Applied Soft Computing 118 (2022) 108439

Contents lists available at ScienceDirect

Applied Soft Computing

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

Cybersecurity of multi-cloud healthcare systems: A hierarchical deep learning approach

Lav Gupta [a,*], Tara Salman [b], Ali Ghubaish [c], Devrim Unal [d], Abdulla Khalid Al-Ali [e], Raj Jain [c]

[a] *Department of Computer Science, University of Missouri-St. Louis, St. Louis, USA*
[b] *Department of Computer Science, Texas Tech University, TX, USA*
[c] *Department of Computer Science, Washington University in St. Louis, St. Louis, USA*
[d] *KINDI Center for Computing Research, Qatar University, Doha, Qatar*
[e] *Computer Science and Engineering Department, Qatar University, Doha, Qatar*

ARTICLE INFO

Article history:
Received 11 July 2020
Received in revised form 25 December 2021
Accepted 5 January 2022
Available online 12 January 2022

Keywords:
Cloud networks
Edge clouds
Network function virtualization
Critical healthcare
Deep neural networks
Stacked autoencoders
Hierarchical neural networks
Multi-cloud systems

ABSTRACT

With the increase in sophistication and connectedness of the healthcare networks, their attack surfaces and vulnerabilities increase significantly. Malicious agents threaten patients' health and life by stealing or altering data as it flows among the multiple domains of healthcare networks. The problem is likely to exacerbate with the increasing use of IoT devices, edge, and core clouds in the next generation healthcare networks. Presented in this paper is MUSE, a system of deep hierarchical stacked neural networks for timely and accurate detection of malicious activity that leads to alteration of meta-information or payload of the dataflow between the IoT gateway, edge and core clouds. Smaller models at the edge clouds take substantially less time to train as compared to the large models in the core cloud. To improve the speed of training and accuracy of detection of large core cloud models, the MUSE system uses a novel method of merging and aggregating layers of trained edge cloud models to construct a partly pre-trained core cloud model. As a result, the model in the core cloud takes substantially smaller number of epochs (6 to 8) and, consequently, less time, compared to those in the edge clouds, training of which take 35 to 40 epochs to converge. With the help of extensive evaluations, it is shown that with the MUSE system, large, merged models can be trained in significantly less time than the unmerged models that are created independently in the core cloud. Through several runs it is seen that the merged models give on an average 26.2% reduction in training times. From the experimental evaluation we demonstrate that along with fast training speeds the merged MUSE model gives high training and test accuracies, ranging from 95% to 100%, in detection of unknown attacks on dataflows. The merged model thus generalizes very well on the test data. This is a marked improvement when compared with the accuracy given by un-merged model as well as accuracy reported by other researchers with newer datasets.

1. Introduction

The cost of providing healthcare is steep and spiraling globally. The US alone spent about $4.1 trillion in 2020, a staggering 19.7% of the GDP [1]. Despite substantial healthcare budgets, most countries are saddled with inefficient healthcare systems. Administrations are frequently blamed for inadequate response to medical emergencies, delays in diagnosis of acute cases, insufficient monitoring of chronic patients, re-admissions, and above all, too many preventable errors. Some readers will be surprised to know that preventable errors are a leading cause of deaths in the US [2].

To improve treatment outcomes with timely and accurate diagnosis and, at the same time, reduce the rising cost burden of healthcare on governments, modern healthcare is increasingly relying on technology. Trends show the prevalence of the Internet of Things (IoT) devices for the collection of patient data and the use of multi-cloud computing for storage and analytics. Cloud adoption, including multi-cloud and hybrid cloud, is projected by International Data Corporation, a market intelligence company, to be a staggering 90% of mid-size and large size organizations by 2022 [3]. In healthcare the growth multi-cloud is expected to be 37% in 2021 as against 19% in 2019 [4]. These developments have the potential of saving crucial minutes in the diagnosis and treatment of critical, hospitalized, or ambulance bound patients. In other words, they could drastically improve the chances of patients' survival and return to good health.

* Corresponding author.
E-mail address: lavgupta@wustl.edu (L. Gupta).

https://doi.org/10.1016/j.asoc.2022.108439
1568-4946/© 2022 Published by Elsevier B.V.

Medical IoT is helping implement automatic remote monitoring and recording of patients' vital signs, allowing closer monitoring, which is not possible manually. The global Covid-19 pandemic has accelerated the adoption of medical IoT in hospitals [5]. Because of rapid upsurge in the use of IoT devices, analytics and storage are being increasingly pushed from premises to the edge and further to large public clouds (also called core clouds in this paper). These technology trends promise to reduce reliance on brick-and-mortar healthcare infrastructure and make the next generation healthcare less expensive and more efficient. Despite the expected gains from the infusion of new technologies, the undesirable fallout is the increasing susceptibility of patients and hospitals to crippling malicious attacks [6]. Over the last two years, 90% of healthcare organizations have suffered at least one cyberattack [7]. This adds up to a 71 percent increase in breaches or incidents in 2020 over 2019 [8]. Among all major sectors of the economy, the largest number of ransomware related insurance claims during 2015–2019 came from the healthcare sector [9]. Any attack on critical patient data, flowing through an intricately designed healthcare system, poses severe threats to patients being diagnosed, treated, or carried to a medical facility on an ambulance. Research of Choi and Johnson shows that data breaches affect a hospital's 30-day mortality rate adversely [10]. These attacks not only threaten serious repercussions on routine medical procedures, but also present the risk of critical medical devices, such as pacemakers, being hacked and endangering patients' lives. A recent example is recall of insulin infusion pumps by the US Foods and Drugs Administration because of the possibility of attackers changing settings and takeover control of insulin delivery [11].

The next generation healthcare is expected to make extensive use of sensing devices, clouds, and virtualization of communication. This makes the system complex, opens the boundaries of the constituent domains and increases the attack surface. Security and privacy concerns have so far limited the adoption of cloud infrastructure in healthcare to about 14% [12]. Conventional intrusion detection systems (IDSs) become largely ineffective in such an environment [13]. Additionally, there is lack of research in finding more effective means to tackle increased risk of attack in the next generation healthcare. This provides the motivation for this work. Consequently, the objective of this work is to create a credible defense, against known and unknown or "zero-day" attacks.

In this paper, we propose MUSE (Merged Hierarchical Deep Learning System with Layer Reuse for Security), a system consisting of a hierarchy of distributed deep learning models in the edge and the core clouds. MUSE secures healthcare providers' operational systems from attacks arising within or outside their organizations. MUSE predicts attacks on dataflows by examining the data being transferred to and from the clouds and detecting even very subtle changes in the metadata associated with these flows. The models working in the edge and core clouds are trained to identify malicious activity by detecting abnormal variations in the metadata features like source- or destination-byte-count, retransmissions because of payload error, mean packet size, content size of the data and packet transfer rate.

The novelty of this work lies in tackling the increased attack surface of the next-generation health care systems by using a hierarchy of cooperating neural network models in the edge and the core clouds. The challenge in doing this is obvious. The distributed deep learners grow in size and complexity as we move from the edge of the IoT domain through the smaller edge clouds to the larger core clouds. Any effort to increase the training speed often results in reduction in detection accuracy. We tackle the dual challenges of fast training of large and complex core cloud models and high accuracy of detecting known and unknown attacks by using an innovative merged deep neural network (DNN) models in the core clouds. The novelty of the method lies in its hierarchical implementation and use of merged trained edge models to reduce training time in larger core cloud while still maintain high accuracy of detection.

Specific contributions of this paper are:

(a) Establishing a reference architecture for critical healthcare applications, e.g., ambulance bound critical patients.
(b) Based on the attack surface presented to the flow of data in the multi-domain next-generation healthcare architecture, evolving a threat model to clearly defne the information technology and operational technology related threats and their mitigations.
(c) Proposing a novel method based on hierarchical DNNs to protect the patient data flowing between the IoT domain and the edge cloud and between the edge clouds and the core clouds.
(d) Evolving a merged core cloud DNN for improvement in training time and accuracy of prediction of abnormal activity in the data flows.
(e) Evaluating the proposed method and discussing the results.

The rest of the paper is organized as follows. In Section 2, we discuss the related work to show how our work fills an existing gap. Section 3 presents the conceptual layout and architecture of the next generation healthcare network, laying down a foundation for discussing the threat model in Section 4. The merged hierarchical model is discussed in Section 5. In Section 6, we discuss the evaluation results. Finally, Section 7 gives a summary of the conclusions.

2. Related work

To get an assessment of the state of the art, we present here selected published research done mainly during 2018–21. Some earlier research might have been included for its contemporary relevance and comparative value.

2.1. Works comparing shallow and deep neural networks for network intrusion detection

During the last three years a number of researchers have published comparison shallow machine learning and deep learning methods for security applications. Gradual improvement of detection capabilities with deep learning are in part responsible for more researchers looking at it as a credible rival to shallow learning for such applications. Kim and Gofman (2018) use NSL-KDD dataset and obtain peak performance of 98.50% detection accuracy for a 17-hidden node shallow network and performance of deep network at 48.30% detection accuracy [14]. This is typical of initial research in application of deep learning in cyber security. Another contemporary research by Xin et al. (2018) finds that both shallow and deep learning have their own advantages and disadvantages in implementing IDSs and none can be recommended over the other [15]. Farahnakian and Heikkonen (2018) show that sparse autoencoders (SAEs) with Support Vector Machine (SVM) for multi-class classification give 84.86% accuracy with the NSL-KDD dataset, which is better compared to a stand-alone SVM model, which gives an accuracy of 79.42% [16].

Nguyen et al. (2018) use RBMs with NSL-KDD, KDDCup99 and UNSW datasets and obtain 90.99%, 95.84% and 97.11% accuracy. The corresponding numbers for SVM are 88.32%, 93.38% and 96.74%. Generalization is weak, and the method is not effective in detecting unknown attacks. They conclude that deep learning methods perform better with large volumes of data and use of high-performance machines with GPUs but have a longer training time than machine learning methods [17]. Nassif et al. (2021) conclude that deep learning outperforms shallow machine learning in many applications, but this may not always be the case for cybersecurity [18]. They experimentally compare the performance of Random Forest (RF) (Shallow Learning) and Fully Connected Feedforward Neural Network FNN (FNN) (Deep Learning) and find that RF performed better with an F1-score of nearly 0.8, against the 0.6 obtained by the FNN. They, however, agree that deep learning for cybersecurity is under-researched.

2.2. Recent work using deep learning for healthcare systems

Though we have not come across any work that studies application of deep learning in security of multi-cloud healthcare systems, there has been obliquely related research that researchers may find useful for healthcare environment. Hayyolalam et al. (2021) present a framework that uses deep reinforcement learning (DRL) in the edge cloud to offload the tasks from the IoT sensors to the edge cloud [19]. Authors claim that the DRL method allows the DRL layers or devices in the edge cloud to work in parallel and help reduce network latency congestion.

Elayan et al. (2021) propose digital twin technology in which a virtual replica of a physical asset is used to diagnose and detect heart problems using a novel Electrocardiogram (ECG) classifier [20]. Each patient has a DT that is updated by this classifier using DTs of similar patients. This allows physicians to be able to predict the outcome of their treatment. Their method uses multiple deep learning methods such as Long short-term memory (LSTM) and Convolutional Neural Network (CNN) with accuracy rates of 97.09% and 96.67%, respectively.

Zaman et al. (2020) have developed a graph theory concept that minimizes the healthcare network drop during rush hours [21]. This concept uses powerful nodes called parent nodes (PNs) that can share their services with other nodes by splitting the tasks between these nodes and one of the PNs as needed. Similar to [21], AL-KHAFAJIY et al. (2021) introduce Cognitive Fog (CF) model that uses multiple CF nodes to offload the tasks from the IoT devices and allow parallel processing on the fog cloud [22]. They use multiple machine learning classifiers, namely Decision Trees (DT), K-Nearest Neighbors (KNN), and Density-based spatial clustering of applications with noise (DBSCAN), to detect anomalies in the transmitted data. The final decision is based on a majority vote of the decisions made by these classifiers. However, using these classifiers requires a lot of labeled data, which is not readily available in healthcare systems.

2.3. Recent application of deep learning methods in diverse areas

Hizal et al. (2021) use a CNN model-based IDS running on a GPU to achieve 99.86% accuracy for 5-class classification using NSL-KDD dataset [23]. Chen and co-researchers (2020) have tested the CNN model with CICIDS2017 dataset and get an accuracy of 96.55% on raw data [24]. Unlike our work the authors use a single flat cloud structure. Training times of the models in the cloud have been discussed.

Gopalakrishnan et al. [2020] present a deep learning based traffic prediction with a data offloading mechanism with cyber-attack detection (DLTPDO-CD) technique [25]. The proposed model involves three major processes: traffic prediction, data offloading, and attack detection. For the attack detection part a deep belief network (DBN) optimized by a barnacles mating optimizer (BMO) algorithm called BMO-DBN is applied for cyber-attacks in mobile edge computing. With BMO-DBN the authors get an accuracy of 97.65%. For comparison they found with DBN they get a lower accuracy of 96.17%.

In the experiments in [26] Xun et al. (2021) utilize CNN and LSTM networks to build different driving behavior evaluation models in edge network assisted vehicle driving. The accuracy rate and loss value of training data in CNN are 96.7% and 0.189, and the value in LSTM are 98.5% and 0.029. On the test dataset, accuracy rates are 90.2% and 95.1% for CNN and LSTM respectively.

Lin's research (2018) shows that methods relying on attack patterns and risk assessment have low accuracy and are not useful in real-time cloud systems. In multi-cloud computing environment, using a combination of restricted Boltzmann machine (RBM) and SoftMax, they achieve the highest accuracy of 95.84% with KDD Cup '99 and NSL-KDD datasets, which is better than machine learning algorithms in similar situations [27]. Otoum et al. (2019) present what they call a behavior-classification approach for network intrusion detection [28]. They present the revised Le-Net 5 CNN model designed by LeCun et al. in 1998. With KDDCup99, the prediction accuracy for major types of attacks achieved is 99.65% with eightfold cross-validation ($>$10,000 records). For attacks with less than 500 records, the average accuracy is 95.41%. Roy and Cheung (2018) present a study on the feasibility of deep learning in a 20 sensor wireless sensor networks [29]. With Bi-directional Long Short-Term Memory Recurrent Neural Network (BLSTM RNN) model and UNSW15 dataset, the authors achieve 95% or more in attack detection. Using RBM based clustered detection system, they achieve an accuracy of 99.91%, with a detection rate of 99.12%. Parampottupadam and Moldovann (2018) have developed a cloud-based prototype system to investigate the capability of deep learning based binomial and multinomial models to detect network intrusions in real-time. They have compared deep learning models built with H2O and DeepLearning4J with other machine learning models like Random Forest, Support Vector Machine, Logistic Regression and Naive Bayes using NSL-KDD dataset. The H2O deep learning based binomial and multinomial models generally outperformed the other models, achieving over 99.5% training accuracy using cross-validation and over 83% accuracy on the test dataset.

Shone et al. (2018) propose a deep learning classification model constructed using stacked Non-Symmetric Deep Autoencoders (NDAEs) [30]. They have implemented the model in GPU enabled TensorFlow and evaluated it using the KDD Cup '99 and NSL-KDD datasets. Accuracies of NDAE and DBN with Kddcup99 are 97.85% and 97.9% respectively. With NSL-KDD the accuracies are 80.58% and 85.42%. The authors have also showed that for training an eight-layer model their NDAE takes 2024 sec as against 54 660 s taken by an eight layer DBN 54660, NDAE 2024 s

Vinayakumar et al. (2019) combine Network Intrusion Detection System (NIDS) and Host Intrusion Detection System (HIDS) to create a deep learning approach based on deep neural network (DNN) to proactively detect and classify unforeseen and unpredictable cyberattacks [31]. The experiments were run for 1000 epochs with learning rate varying in the range [0.01–0.5]. For older datasets like KDDCup (1999) most of the DNN network topologies showed train accuracy in the range of 95% to 99% while the newer UNSW-NB15 (2015) and WSN-DS (2016) showed lower train accuracy in the range of 65% to 75%.

The work of Abusitta and co-researchers (2019) is one of the very few on multi-cloud but involves use of non-hierarchical cloud structure for IDS [32]. Each of the cloud models have incomplete information and work cooperatively. Normally receiving feedback from various clouds and aggregating them takes time and make the models unsuitable for real-time applications. The authors propose a Stacked Deionizing Autoencoder Model (SDAE) model that makes decision in the absence of complete feedback from the IDSs. It has been implemented on GPU-enabled TensorFlow and evaluated using a dataset derived from KDDCup99 dataset. The average accuracy obtained by the proposed model at different numbers of hidden units (ranging from 70 to 350) is 87.5%.

He et al. (2021) have worked on CNN based supervised pre-training module and the AE-based data reconstruction module. With the USTC2016 datasets they achieve 97.4% and 97.95 for malicious and benign traffic respectively and with CIC-IDS2017, 89.45% for malicious and 80.25% for benign traffic [33].

In another recent study, Udhendran and Balamurugan (2021) use autoencoder for farming applications with a year 2020 dataset from Plant Pathology, to achieve an accuracy of 95% for training data and 90% or more the validation data (2021). The work by Elayan and others (2021) deals with detecting diseases and problems from the ECG DT using MIT-BIH Arrhythmia Database. With LSTM the authors achieve validation accuracy of 97.09% and training accuracy of 98.96%. Using CNN validation accuracy is 96.67% and training accuracy is 98.96% [20].

In [19] Hyolalam et al. (2021) propose a smart healthcare framework that makes use of edge technology and deep reinforcement learning. Processing takes place at the edge but can be offloaded to public cloud if the volume of the data so demands. The authors have not performed any experimental evaluation for detection accuracy or computational complexity but feel that for real-time health applications some method of training reduction would be necessary.

From the above-mentioned findings, we can conclude that there have been sporadic successes in applying deep learning to network intrusion detection. However, such work remains in its infancy [34]. We did not come across any work that uses a merged model for hierarchical distributed networks in multi-clouds. The work in [35] introduces elements of the initial work on hierarchical neural networks in the context of critical services. However, the concept has been developed, tested and described for the first time in more detail this paper.

3. Conceptual layout and architecture of the next generation healthcare system

The generic layout of the next generation healthcare service assists in understanding the flow of data and provides useful inputs for evolving the system architecture. This architecture is then used to prepare a threat model and carry out mitigation planning specific to the cyber–physical system that we are considering. We depict the overall layout of the next generation of healthcare services in Fig. 1 and describe its domains below. The constituent domains act as a source, a sink, a storage or an analytics resource.

3.1. The healthcare network domains

As shown in Fig. 1, we subdivide the network into the IoT, the multi-cloud domain and the visualization domains. The data transmission among the domains can use contemporary wide area network services or virtual network service (VNS) to which the carriers and Internet Service Providers (ISPs) are expected to increasingly migrate. A brief description of these domains is given below:

(a) The IoT domain: The IoT domain consists of a heterogeneous mix of wired and wireless, wearable, ingested, or implanted biosensors, actuators, and other medical devices, for patient data acquisition and delivery of treatment. Each of the sensing devices performs simple tasks like monitoring pulse rate, oxygen saturation, or blood pressure. Actuators, on the other hand, perform the task of delivering treatment like activating oxygen flow or injecting insulin, based on commands from clinical staff or other devices. In the case of ambulance bound patients, because of monitoring by IoT devices, there also are auto-responses in terms of alerts and suggestions to the paramedic staff inside the ambulance. This domain is generally a source, producing large volumes

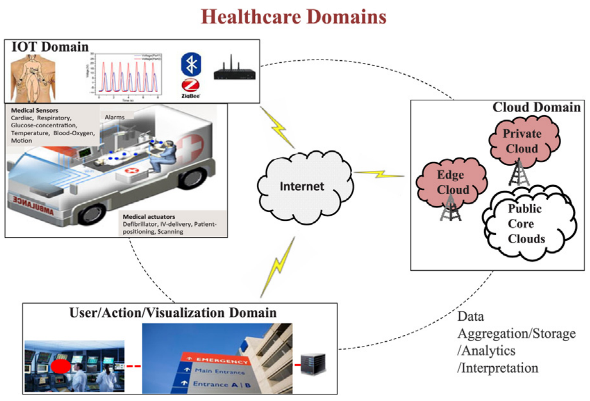

Healthcare Domains
IOT Domain
ZigBee
Medical Sensors
Cardiac, Respiratory, Glucose-concentration, Temperature, Blood-Oxygen, Motion
Alarms
AMBULANCE
Medical actuators
Defibrillator, IV-delivery, Patient-positioning, Scanning
Internet
Cloud Domain
Private Cloud
Edge Cloud
Public Core Clouds
User/Action/Visualization Domain
EMERGENCY
Main Entrance
Entrance A | B
Data Aggregation/Storage /Analytics /Interpretation

Fig. 1. The conceptual layout of next-generation healthcare.

of multi-dimensional patient data, but also acts as a sink for the commands sent to the actuators and other devices.

IoT devices present several operational challenges due to constrained battery power, small memory, and low processing capabilities. Consequently, these devices depend on external storage, processing, and analytics provided by the multi-cloud domain. At the edge of the IoT domain is the IoT gateway to which most of the devices connect. The IoT gateway also acts as an interface between the IoT domain and the cloud domain and does the task of protocol or data conversion when required. Some of the devices may connect directly to the service providers wired or wireless VNS.

(b) The cloud domain: In our work, the cloud domain has been assumed to consist of the hierarchy of edge and core clouds, as shown in Fig. 1. The edge clouds will be like those provided by the mobile service providers through the computing equipment on mobile towers collocated with the base stations. These edge clouds are closest to the patient and provide relatively inexpensive and low latency connectivity to the multi-cloud system. The edge provides intelligence for all the workload that can be better performed by cloud resources closest to the devices in the IoT domain. The larger core clouds are public clouds providing ample storage for historical data and more sophisticated analytics. The data collected by the edge clouds, may usually be required for diagnostics in the current acute ailment of patients or active monitoring of patients in the edge cloud area. When no longer actively required in this manner, the data can be moved to the core clouds for permanent storage. The core clouds can help analyze a vast amount of patient historical data (PHD) and real-time data (RTD) using sophisticated AI based analytics. However, the connectivity cost and latency will be much higher than that of the edge clouds. In this paper, we will use the terms core cloud and main cloud interchangeably with the name public cloud. When arranged in a hierarchical manner, a multi-cloud design provides a combination of low latency, large storage, optimized bandwidth cost, and high analytical sophistication. While the edge clouds aid quick diagnosis in emergency and acute cases, computationally demanding inference algorithms in the larger core clouds help in determining patterns in historical data that correlate with current symptoms. This helps in differential diagnosis or discovering yet to surface ailments [36].

(c) The visualization domain: The visualization domain is predominantly a sink that consumes the analyzed data in many forms. The clinical staff using the data can also generate small amounts of data in the form of commands, instructions, or prescriptions. This domain consists of mechanisms for presenting multiple streams of processed data to the concerned clinical staff. The domain also allows the clinical team to choose from several streams

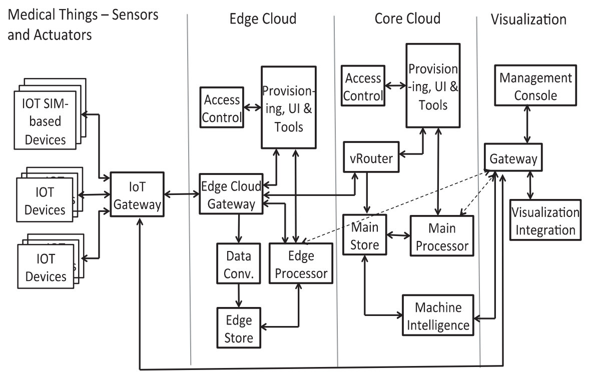


Fig. 2. Architecture of the IoT-Cloud healthcare system.

of incoming information to get a good idea of the health of the patient, catch the exceptions and anomalies, and the early signs of developing complications [37]. The information can be presented in graphics, tabular and other forms, to assist doctors in making a fast and accurate diagnosis. In the case of data from an ambulance, the doctors may decide to communicate with the paramedics to provide guidance for immediate patient care. It is understood that improvement in visualizations can improve diagnosis and, in turn, outcomes of the treatments.

3.2. The architectural design of next-generation healthcare

The architecture is shown in Fig. 2. The functional aim of the design is to provide comprehensive data to medical experts as well as applications that process them or control the devices used for patient care. This requires functionalities like image acquisition, analytics, and communication, remote patient consulting, continuous monitoring, and telemedicine. All this results in the movement of a large amount of data from the IoT domain to the cloud domain for analytics and storage.

The proposed architecture is flexible and scalable; it grows with the demand on the system and yet retains its amenability to effective security. To maintain consistency with the layout already presented in Fig. 1, the architecture has been divided into four inter-linked domains: the sensor and actuator domain, the edge cloud domain, the core cloud domain, and the visualization domain. This compartmentalization into domains assists in modular design of the security policy for the entire healthcare network. A brief discussion of the essential constituent of various domains follows:
(a) IoT gateway: This gateway will be present in ambulances, hospitals, offices or homes. They link the IoT domain and the Edge cloud domain. Other devices in the domain rely on their connectivity with the IoT gateway for many functions like protocol and data conversion, and connectivity to cloud gateways. There can be more than one IoT gateways each connected to their own set of devices. The IoT gateways connect to the service provider's virtual or physical wide area network and can register with the edge clouds or directly with the core cloud. Some of the devices that are mobile-SIM based can directly connect to the communication network and can securely register with the cloud for sending and receiving data. The gateways assist in device provisioning, data filtering, batching and aggregation, buffering of data, protocol translation, event rules processing, and more. There can be more than one IoT domain with their own gateways connected to the edge clouds.
(b) The edge and core cloud gateways: These gateways are in the edge and core clouds of the multi-cloud domains. The cloud gateways may provide data compression for faster transfer, multiple bandwidth options and distribute workload. They provide connection to the IoT gateways and devices and collect telemetry information. These gateways may be physical or in the form of virtual routers (vRouters).
(c) The edge and main processors: These processors work on the streams of data from the devices or the gateways. The edge processor has reasonable processing capabilities and may have specialized hardware like the edge tensor processing unit. It has capabilities to train and use small neural network models. The edge store is primarily for temporary storage to keep the patients' data while the ambulance is passing through the corresponding cell or while it is needed for patients' present care. The main (core cloud) processor can carry out more sophisticated analytics based on large neural network models using both current data and large historical datasets. It has the capability of interfacing multiple streams of heterogeneous information with the visualization domain.
(d) Machine intelligence agent: In addition to the main processor, the core cloud domain may provide specialized hardware (e.g., GPUs) and software that can process historical data for intelligence on developing diseases or for predicting re-admissions.
(e) Visualization: Visualization domain consists of related tools used to visualize patient data and facilitate patient diagnosis, monitoring, and management. It consolidates and synthesizes large volumes of data from multiple sources to provide critical insights to the clinical staff. It makes patterns and relationships evident in large amounts of data, which are not discernable in raw data or reports [38].
(f) Access control: This provides various forms of authorization, authentication, and accounting of connections attempted by people and devices. It prevents unauthorized agents from accessing stored or analyzed information from the edge or the core cloud zone.
(g) Provisioning, UI, and other tools: These tools give the domain manager a user-friendly interface to inject policies and provision cloud resources.

3.3. Security architecture for the next generation healthcare

The overall security architecture that has been taken into consideration is shown in Fig. 3. The next-generation healthcare systems will have security built-in by design rather than being overlaid. This reduces manual provisioning and allows security to be consistent across all domains and leads to more robust end-to-end security [39]. It also ensures high availability, patient safety, conformance to regulations and improved domiciliary and ambulance-bound patient care. The methods that we have evolved are consistent with the described security architecture.

In this work, we have focused on the development of a hierarchical deep-learning-based anomaly detection system that will indicate if the dataflow between the IoT gateway and the edge cloud or between the edge cloud and the core cloud has been compromised. The relevant aspects are discussed below:
(a) User authentication: Patients, paramedics in the ambulances, hospital medical and support staff all have to be authenticated securely and robustly to protect the patients and related data. For patients in the ambulance, brain wave biometrics can be explored. Cryptographic authentication and authorization techniques combined with biometrics can provide a high level of system access security.
(b) Data at rest in the cloud: Confidentiality of data stored at various points in the network can be protected through innovative encryption and encryption key management. The use of innovative hashing algorithms can safeguard the integrity of data.

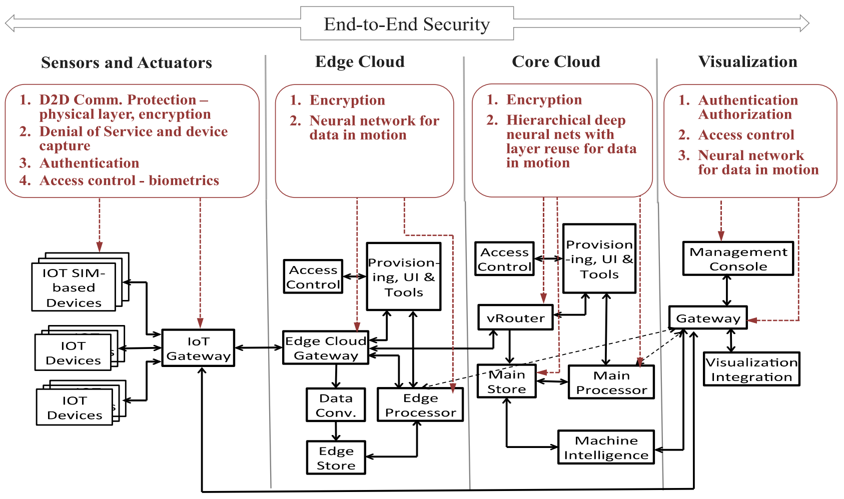

End-to-End Security
Sensors and Actuators
Edge Cloud
Core Cloud
Visualization
1. D2D Comm. Protection – physical layer, encryption
2. Denial of Service and device capture
3. Authentication
4. Access control - biometrics
1. Encryption
2. Neural network for data in motion
1. Encryption
2. Hierarchical deep neural nets with layer reuse for data in motion
1. Authentication Authorization
2. Access control
3. Neural network for data in motion
IOT SIM-based Devices
IOT Devices
IOT Devices
IoT Gateway
Access Control
Provision-ing, UI & Tools
Edge Cloud Gateway
Data Conv.
Edge Processor
Edge Store
Access Control
Provision-ing, UI & Tools
vRouter
Main Store
Main Processor
Machine Intelligence
Management Console
Gateway
Visualization Integration

Fig. 3. The security architecture of the next-generation healthcare.

Public key cryptography can be used to provide both integrity and confidentiality. In time-sensitive systems, the latency introduced by these techniques needs to be monitored.

(c) *Data in motion:* This is the part that is of interest in this paper. Data from the physiological sensors are collected by the IoT gateway and transmitted over the Internet or wireless access network to the edge cloud. Low latency tasks are performed here, and for a more in-depth analysis, the data is transferred to the core cloud. The processed information is sent to the doctor's screens to visualize the exceptions and make an accurate diagnosis.

The architecture explained above helps to realize our aim of protecting data in motion from any attack that threatens to affect the meta-information or payload of the dataflow. As explained in Section 1, changes in the payload affect features that are part of the meta-information. The main concern here is to *use an innovative deep learning technique to find anomalies in the inter-domain streams of data for any indication of malicious intent.*

4. Threat model for the healthcare network

The threat model essentially provides an understanding of the attackers, attacks, and mitigation and is crucial in deciding the security strategy. In IoT-Multicloud systems the designers must rely on the convergence of *information* technology (IT) and *operational* technology (OT) to run and maintain infrastructures. The architecture and the threat model that we propose draw from the Microsoft STRIDE architecture for clouds, covering threat categories Spoofing, Tampering, Repudiation, Information disclosure, Denial of service and Elevation of privilege, has been widely applied to cyber–physical system and is applicable to our situation with suitable changes [40]. This model recognizes the trust boundaries and the vulnerabilities that come with transgressing domains in the IoT-Multicloud systems. In critical operational technology it is not only the data leak that is to be worried about but also how theft and manipulation of data may impact safety of the stakeholders. In situations, as presented by Covid-19, when the system is overstretched, vulnerabilities could have serious ramifications [41]. The compartmentalization or zoning feature of our architecture has helped us formulate an appropriate threat model. The gateways demarcate the domains. Thus, we have the IoT gateway, the edge-cloud gateway, the core cloud gateway, the visualization gateway with their associated services. A trust boundary separates each domain from the other connected domains. It is vital that the data flow authorization techniques

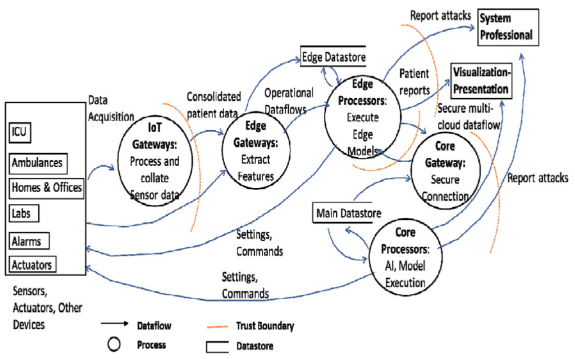

Report attacks
System Professional
Edge Datastore
Patient reports
Visualization-Presentation
Data Acquisition
Consolidated patient data
Operational Dataflows
Edge Processors: Execute Edge Models
Secure multi-cloud dataflow
ICU
Ambulances
Homes & Offices
Labs
Alarms
Actuators
IoT Gateways: Process and collate Sensor data
Edge Gateways: Extract Features
Core Gateway: Secure Connection
Report attacks
Main Datastore
Settings, Commands
Core Processors: AI, Model Execution
Settings, Commands
Sensors, Actuators, Other Devices
Dataflow
Process
Trust Boundary
Datastore

Fig. 4. DFD for healthcare dataflows and trust boundaries.

combined with biometrics can provide a high level of system access security. The data crossing the trust boundaries should be protected against various kinds of attacks [42].

A data flow diagram (DFD) is an important tool to represent processes (entities like gateways can be abstracted as processes), data stores (like the permanent store in the core clouds that stores patients' historical data), data flows (between the IoT, edge and core clouds being of concern) and entities that interact with the system. Fig. 4 shows a diagrammatic representation of these elements in a DFD. Each of the elements in DFD are subject to all or some aspects of STRIDE.

4.1. Attack surfaces

The attack surfaces at various gateways are described below:

(a) The IoT gateway attack surface: The IoT gateways are in places like ambulances, hospitals, offices, and homes. We can see from the data flow in Fig. 4 that these gateways are at the perimeter of sensors and actuators domain. An IoT gateway may combine the functionalities of a router, a processor, a protocol and data converter, a device controller, a data flow manager, and a sessions manager. There are two attack surfaces one facing the IoT devices and the other facing the edge cloud zone. These gateways are often targeted for intrusions. These intrusions can have severe effects as there is no redundancy built into these gateways.

(b) The edge-cloud gateway attack surface: The edge cloud gateways are instantiated on physical- or virtual-servers. They accept flows from one or more IoT gateways (in some cases directly from

the devices) and regulate access to the facilities of control, analysis, and storage in the edge clouds. A mobile edge cloud gateway is usually near the cellular tower and, therefore, physically less accessible. The gateway has two attack surface areas, one facing towards the IoT gateways and the other towards the core cloud gateway.

(c) The core-cloud gateway attack surface: The core cloud gateway controls access from the hospital/ambulance zones coming directly or through an edge cloud gateway. It also connects the visualization domain and other external entities of the healthcare system like insurance agencies, pharmacies, suppliers, and billing services. Both the edge and the core clouds usually have interconnection with the Internet making the cloud domain more vulnerable.

4.2. Attackers and attacks

(a) Attackers: These are internal or external malicious agents who attempt to mutilate or alter the flow of data in any way. The threat could be from organized crime, nation-states, hacktivists, business associates, skiddies, and malicious insiders. In this threat model, we assume that adversaries have unlimited resources and time, while the targeted organization has limited resources, and thus, needs efficient methods to counter attacks [43].

(b) Attacks: Attacks that target confidentiality, integrity, and availability (CIA) are major security issues for cloud based IoT. Adversaries could maliciously exploit several vulnerabilities in many forms. They can carry out a variety of attacks like DOS to prevent patients' access to devices, changing device settings, or stealing patient records for frauds like undergoing surgery or obtaining prescription medications. More severe attacks include capturing patient devices, ransomware, and advanced persistent threat (APT). The attackers, also referred to as cybercriminals, evade detection by using multiple tactics, tools, and targets in their attacks. Their techniques can change temporally and spatially.

It is essential to recognize the kind of attacks that we are protecting the system against. In our system, we are concerned with any intrusion that alters the dataflow in terms of the meta-information or payload. This will happen if the dataflow is disrupted in any way. In our DFD of Fig. 4, the processes, dataflows and datastore are subject to the following threats:

(i) *Spoofing: This* is a man in the middle attack where the intruder fools the source to believe that it is a legitimate destination. Spoofing attacks can help attackers to cross the trust boundaries and cause data theft or deletion. If the masquerader intercepts, partially or fully modifies the dataflow, then this is of concern to our system. A phishing attack is a glaring example of spoofing that can target healthcare systems.

(ii) *Tampering:* Tampering is a serious infringement to the healthcare system in which an attacker alters the dataflow and thereby changes the values of the patients' biomarkers or the meta-information of the dataflow (e.g., packet flow rate). Examples of such attacks for the IoT domain include packet injection or packet crafting to change patient metadata. An attacker may tamper with the settings or software of a device, potentially causing it to malfunction.

(iii) *Denial of Service (DoS):* IoT devices are generally constrained in many ways and cannot deploy sophisticated security mechanisms. These devices can be flooded with unsolicited traffic and rendered inaccessible for genuine traffic. If devices are listening for inbound connections, then an attacker may open many connections and not service them. A malicious actor can also spoof a number of these and use them to launch distributed DoS attack against targets in the edge or cloud clouds. In healthcare this may mean that patient devices like a ventilator or control devices for service like oxygen supply can be rendered inoperable.

(iv) *Information Disclosure:* Constrained devices will have simple security like a single PIN or password. Sometimes they just trust the network and allow access devices on the same network. Active reconnaissance may enable the attacker to obtain information about the target and then cause remote attacks. SQL injection attack can cause an information disclosure attack as it can collect information about the data in the system.

(v) *Elevation of Privilege:* A user with limited privilege assumes the identity of a privileged user and gets administrative rights. With higher privileges, the attacker can cause an exploit attack. It can access gateways and change information flows.

(vi) *Repudiation*: An attacker can log data to wrong files or change data in the name of others.

(vii) *Ransomware:* Ransomware attacks on hospitals are increasing and remain a growing concern. Not only they are a financial drain on financially constrained medical facilities but also cause disruptions in provision of optimum healthcare to patients.

(viii) *Advanced Persistent Threat (APT)*: This is an insidious attack in which hackers gain undetected access where they can spend time gaining valuable inside data and understanding of the systems before launching a targeted attack.

In IT-OT hybrid systems like healthcare, consequences of attacks can manifest in many forms [44]. Some of these are described below:

(i) *Loss of control and safety impact*: Loss of control in the IoT domain or any of the clouds could potentially bring the medical activities to a halt and pose a threat to health and safety of people in the affected areas.

(ii) *Financial impact*: There is a recovery cost for restoring the system to the pre-attack state and payment of ransom in some cases. Then there is indirect cost of malfunctioning devices, loss of control of life sustaining systems and declining revenues. There may also be litigation charges, insurance fee and payment of damages to patients.

(iii) *Disruption to operations*: A major cybersecurity breach in OT can severely affect the ability to supply goods and services. This could impact the organization's ability to deliver on its mission and result in the loss of credibility and customers.

(iv) *Loss of trust*: A breach could result in lost reputability and trust from the public, patients, and investors.

(v) *Data exposure*: The loss of data in operational technology settings can expose personally identifiable data of patients, patient medical and financial records and medical researchers' intellectual property.

(c) Mitigation: In the described architecture security is built by design. As far as the dataflows are concerned, we need to take care of both seen and unseen attacks. OT networks within the IoT domain and the cloud network are separated by trust boundaries which needs to be protected. As far as access to cloud resources are concerned, a 'zero trust' policy would usually be employed. Patients and medical professionals are all authenticated when they access these resources. For patients, who are comatose, body's electrical activity may be used for identification. Partner organizations like suppliers, insurance companies, pharmacists, labs may remotely access the IT-OT healthcare system. This further blurs the IT-OT segmentation and expand the attack surface providing new entry points for attackers [45].

This will be discussed in detail in the next section.

5. Merged hierarchical security for data in motion — Theoretical background, methods and procedures

In this section, we discuss our work on a distributed and hierarchical deep learning solution to protect the data flows from any adversarial attack that mutilates or alters the data stream, including its meta-information, in any way [46,47]. Given the multi-cloud hierarchy, we focus on the data in motion into the edge clouds and from edge to the core clouds.

5.1. Design aspects of the security system

We have taken note of the recommendations of International Standards Organization's ISO 27005 in our design. These recommendations describe the process in the following steps — establishing context, identifying and analyzing risk and evaluating and treating them [48]. As far as their functionality is concerned, security systems are expected to monitor and report intrusions continuously. It is also important to have low false negatives so as not to miss out on serious attacks and low false positives to avoid procedural expenses on false alarms [49].

In our work we use proven techniques of threat analysis and vulnerability assessment based on IT-OT philosophies and use deep learning to create a hierarchical multi-cloud system that would meet our aims set forth in Section 1. Keeping in view that a centralized IDS in a particular gateway would have a limited zone of protection, we have decided on a distributed solution that works in the IoT gateway, edge clouds, core clouds. The proposed system incorporates novel techniques to speed-up training and retraining of complex cloud models working on high-dimensional and high-volume data. This makes the system suitable for real-time and near-real time applications.

5.2. The choice of the deep learning system

In our target IoT-Cloud-VNS architecture, the system that we propose has to deal with numerous markers and interaction patterns across a large number of sensors and servers constantly during operation. The data produced by such a system is multi-dimensional, voluminous, and contains underlying patterns that do not become evident with traditional statistical analysis. Researchers have applied machine learning to security in healthcare systems in many forms [49]. Classifiers like SVM, Decision Trees (DTs), Naïve Bayes, K-Nearest Neighbors (KNN), and Random Forest (RF) have been used. It turns out that classical machine learning is not particularly well suited for the application under consideration because of its inherent complexities. We are dealing with unknown attacks for which training data will not be available. Machine learning models trained only on normal data do not generalize well. These methods also usually have relatively high false-positive rates for detection [34], which, in healthcare systems, may cause the risk of overmedication or unnecessary procedures.

Deep learning, on the other hand, has been found to be capable of handling large volumes of labeled and unlabeled data. It makes use of many layers of non-linear processing to select useful features from a high dimensional dataset and even fuse these features to end up with a rich set of automatically selected features. It can handle missing values and can use metadata effectively. These properties give this class of methods high power to analyze and classify data in which features are related to the outcome in a complex way.

We have seen in Section 2 that deep learning models can perform better than shallow machine learning models [17]. Some recent successes have renewed interest in deep learning [36,50,51]. As these improvements seem to be consistent across a large variety of domains discussed above, we have viewed this as a motivation to use deep learning in the healthcare domain. We tackle the major issue that is faced in using deep learning, that is, the high volume and dimensionality of the training data. We have evolved deep learning techniques that improve training speeds while giving high accuracy.

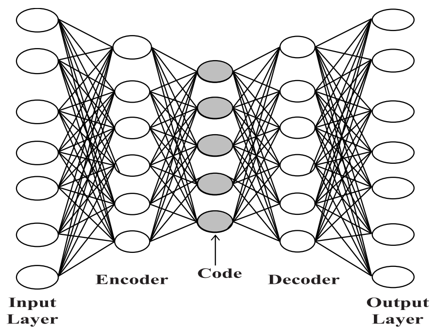


Fig. 5. Representation of a simple autoencoder.

5.3. Deep neural networks — stacked autoencoders

We give a brief description of DNNs with an emphasis on SSAE. For a more detailed treatment, readers are referred to [52,53]. A neural network has an input layer, one or more hidden layers, and an output layer. Each layer is connected to the next layer, fully or partly. When the number of hidden layers is more than one, the resulting network is referred to as a DNN. As the data passes through the layers, the weights of the previous layers are processed using a weight matrix to obtain the weights of the next layer. Each hidden neuron has an activation function (e.g., Sigmoid or Rectified linear unit (RELU)) that determines the activation level of the neuron. The DNN is a feed-forward network (FFN) since the output of one layer is used as input to the next. A loss function, e.g., mean square error (MSE) or cross-entropy, gives the error between the prediction and the actual value. A technique called back-propagation can be used to reduce these errors. The errors are fed back from the output through the hidden layers to the input layer to fine-tune the weights.

We have used a particular type of DNN called autoencoder (AE) shown in Fig. 5. The middle layers, between the input and the output layers, are hidden layers as they are not visible from input or output and are internal to the neural network. The innermost hidden layer has the least number of nodes and is also called the code. The code captures in a condensed form all the intelligence derived from the inputs. Our example neural network in Fig. 5 has 7 input units (bias unit not shown), 5 hidden units, and 7 output units. The output of each layer is directly connected to the input units of the next layer. Each input $x_{i,}\mathrm{x_i} \in \mathrm{R}^n$, goes into a neuron (the computational unit) of the input layer. Output of a neuron is 1 when it is activated and 0 when it is not activated. Activation can be based on a activation function like sigmoid or RELU. The AE is trained to compress into a lower-dimensional code and then reconstruct the output from this representation. The AE tries to learn a function $\mathrm{h_{W,b}(x) = f(W_i x_i + b) \approx x}$ where W are the weights and b the biases.

A sparse autoencoder (SAE) has a sparsity parameter that restricts the code size required for reconstruction. Sparsity enforcement results in only some of the neurons of a hidden layer to be active. In our work, we use stacked SAE (SSAE), which consist of multiple layers of SAEs. This configuration gives rise to deep neural networks that can learn and model non-linear and complex relationships. A representation of the SSAE is shown in Fig. 6.

The pre-training of such an encoder can be done either in a layer-wise manner or end-to-end. In the layer-wise training, we consider each set of two adjacent layers as an AE. We start with the weights $\mathbf{w}$ between the input and the first layer and get $\mathbf{w}^{(1)}$ such that the loss function $L(\boldsymbol{x}, \boldsymbol{z}^{(1)})$ is minimized where $L(\boldsymbol{x}, \boldsymbol{z}^{(1)})$

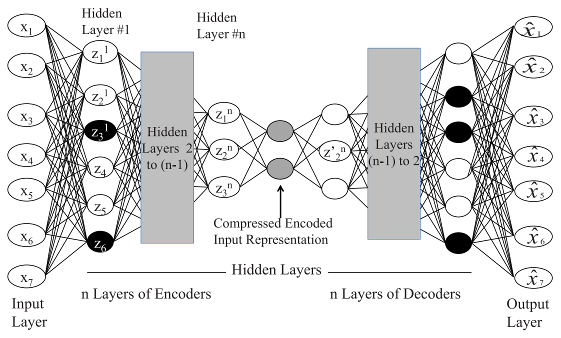


Fig. 6. A stacked autoencoder.

$= \frac{1}{2}||$ $\boldsymbol{x}$-$\boldsymbol{z}^{(1)}||^{2}$ is the MSE. The output at the first hidden layer is $\boldsymbol{z}^{(1)} = f(\boldsymbol{w}.\boldsymbol{x}^{T})$, where the superscript T denotes transposition.

For the second hidden layer, the $\mathbf{z}^{(1)}$ vector becomes the input and $\mathbf{z}^{(2)}$ vector the output. This layer is trained such that the MSE between $\mathbf{z}^{(1)}$ and $\mathbf{z}^{(2)}$ is minimized, i.e., the loss function $L(\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)}) = \frac{1}{2}|| \boldsymbol{z}^{(1)} - \boldsymbol{z}^{(2)}||^2$ is minimized. The output vector $\mathbf{z}^{(2)} = \mathrm{f}(\mathbf{w}^{(1)}.\mathbf{z}^{(1)\mathrm{T}})$. Where the superscript T denotes transposition. This continues until we reach the last hidden layer, which is connected to the output layer.

We continue to produce the output vector like what we have done so far. Thus, for n hidden layers

$$\mathbf{y}' = f(\boldsymbol{w}^{(n)}.\boldsymbol{z}^{(n)T} + \boldsymbol{b}^{(n)}) \tag{1}$$

When some training samples are available in the form of $(\mathbf{x}, \mathbf{y})$, i.e., (feature vectors, output vector), we can use these samples to fine-tune the pre-trained SSAE. We initialize the weights connecting the nth trained layer to the output layer and get predictions $\mathbf{y}'$ and adjust all the weights such that $\frac{1}{2}\ \|y - y'\|^2$ is minimized. Here, $\mathbf{y}$ is the ground truth from training samples $(\mathbf{x}, \mathbf{y})$. This last step is the refinement of all the weights. The overall cost function to be minimized will have a regularization term added to the loss function, which calculates the MSE. Thus, the cost function will take the form:

$$\boldsymbol{J}(\boldsymbol{W},\boldsymbol{b}) = (\boldsymbol{1}/\boldsymbol{n})\sum_{\boldsymbol{i}}(\boldsymbol{MSE}) + \lambda.\boldsymbol{\Omega}_{\boldsymbol{W}} + \boldsymbol{\beta}.\boldsymbol{\Omega}_{\boldsymbol{sparsity}} \tag{2}$$

The first term is the MSE averaged over all the training examples. The second term adds a regularization term on the weights to prevent the regularizer from becoming too small. The coefficient λ determines the influence of this term. The third term is the sparsity regularizer (with coefficient $\boldsymbol{\beta}$), which imposes sparsity constraint on the output from the hidden layer. A commonly used regularization is K–L (Kullback–Leibler) divergence (D_{KL}), which adds a large value if the activation level of a neuron is not at the desired level.

The weights and biases associated with each SSAE are learnable parameters. Some hyperparameters, like the number of layers, the number of neurons and the parameters of the loss function, must be suitably set to obtain good result. Table 1 shows some of these hyperparameters and their typical values in this work.

It is essential to set these hyperparameters carefully such that the innermost compressed layer learns the most useful features and their combinations. For more technical details of the AEs, the readers are referred to [54].

Training is carried out according to the flowchart given in Fig. 7. To make the training process effective, we choose to train one layer of the DNN at a time using unsupervised data. By initializing a network with small random weights (say, uniformly between −0.1 and 0.1), the network is unlikely to fall into a trivial, symmetric local optimum. The reconstruction error indicates whether the data flow is normal or abnormal. Only flows with normal instances are used to train the SSAE. After training, the SSAE will reconstruct information about the normal data flow with low root MSE (RMSE) while failing to do so for anomalous data, which the SSAE has not earlier encountered.

Table 1
Hyperparameters of SSAE.

Parameter	Description	Typical value(s)
Number of layers	Decides the depth of the neural network	4, 12
Number of neurons in the all layers	Decrease from input to the code and then symmetrically increase till the output layer	60, 180
Code size	Innermost layer with most compressed representation of inputs	30
Loss function	In this work we generally use mean square error	
λ	Regularization factor	0.00000001
$\boldsymbol{\beta}$	Sparsity regularizer	0.2
$\boldsymbol{\beta_1}, \boldsymbol{\beta_2}$	Adam optimization decay rates	0.8, 0.9

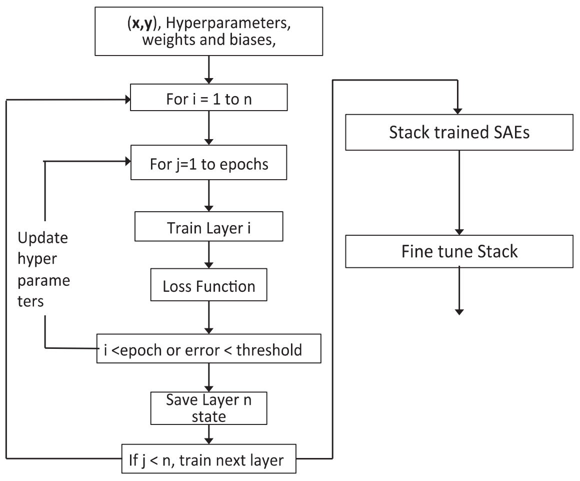


Fig. 7. Layer-wise training of the SSAE.

The dataset is divided into a training dataset and one or more test datasets. A training example is selected from the training dataset, and then the values of output are checked for their quality of reconstruction. If the chosen indicator of error, e.g., RMSE, is consistently below a threshold, then the training concludes.

The trained model is then tested with the test dataset. It is a common practice among the developers of neural network models to "cross-validate" the network on the test dataset periodically during training and to save the network weight configuration meeting one of two criteria: (1) the network with the lowest error on the training dataset or (2) the network with the lowest error on the test dataset. The latter technique is often used to prevent the network from overtraining because networks are prone to overfitting.

The overall duration of training is often expressed in terms of the number of epochs required to reach an error minimum [55]. We will discuss the setting of hyperparameters in the evaluation section.

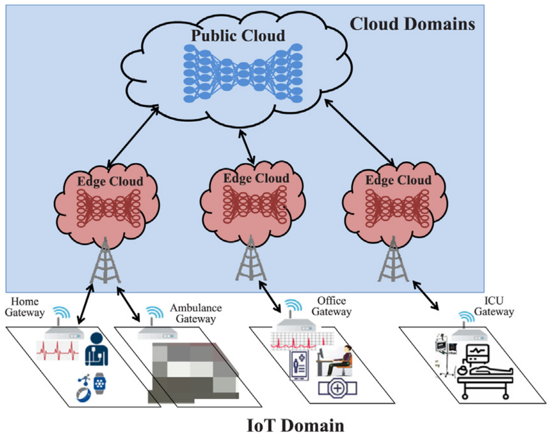


Fig. 8. The hierarchical autoencoder based MUSE model.

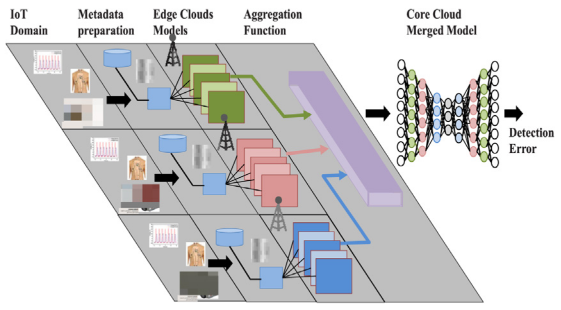


Fig. 9. The merged model.

5.4. MUSE: The hierarchical merged model with layer reuse

The aim of MUSE is to provide security for data in motion from IoT domain to the edge clouds and from the edge clouds to the core cloud. If we consider the security architecture, discussed in Section 3.3, MUSE focuses specifically on the perimeters of the edge and core clouds. Based on our threat model, we look for adversarial intrusions that change data flows, like those caused by exploits, active reconnaissance, SQL injection, denial of service, tampering, device resetting, and device takeover. The main concern here is to use deep learning to see whether the inter-domain streams of data have been compromised, rather than classifying the type of attack. This is based on the meta information extracted from the dataflows [56]. MUSE works in the unsupervised mode and uses the reconstruction error of a data point to differentiate between normal and abnormal data. As explained in the last section, the loss is represented as the residual sum of squares, and the error of reconstruction is represented as the RMSE. The main objective then becomes the reduction of the loss function, or the RMSE by training the model adequately in the forward direction and fine-tuning using techniques back-propagation.

A diagrammatic representation of the distributed hierarchical structure of the SSAE based MUSE intrusion system is given in Fig. 8. In this work, we have restricted our discussion to the data flowing into the edge clouds and the core cloud. The model is backward extensible to IoT gateways and also to private clouds if they exist in the cloud hierarchy.

The use of the concept of merged models in MUSE reduces the training time. A merged model at the core cloud is an aggregation of the edge models done in such a way that the training of layers at the edge can be reused at the core cloud. Fig. 9 illustrates how the edge models are merged to create a core cloud model. The dataflows from the IoT domain passes through the meta-data extractor, which extracts network-flow related features like the packet or byte count, inter-packet times, source, and destination addresses. The aggregator is a construct implemented in software that assembles the deep learning model in the core cloud by combining trained layers from the edge clouds. The aggregator provides the flexibility of taking all the trained layers of the edge clouds or selected layers to optimize the outcomes. A combination of features from data and meta-data are passed through the SSAE at the edge clouds in the service area in which the IoT equipment falls. In conformance with the architecture of the network discussed earlier, the SSAEs are arranged hierarchically with Level 1 or small, 1–2 layer AEs at the IoT gateway, Level 2 or medium, 3–5 layer AEs at edge clouds and Level 3 or large AEs with more than ten layers in the core clouds. These numbers have evolved from simulations using actual IoT network data and health system data generated on the testbed created in the lab at Washington University in St. Louis (Fig. 13) [43].

It is worthwhile to mention that the use of the merged model that allows for reuse of the trained layers is different from transfer learning. The latter primarily uses some well-known trained model for a new problem where the model is of the same type and size. The MUSE model has the flexibility of using parts of the edge cloud SAE and construct an SSAE at the core cloud that will generally be of a different size.

5.5. Design and implementation

The operational basis of the MUSE system is to consume markers from the IoT domain, which generates data from sensors and other devices 24 $\times$ 7. This domain consists of patients being monitored in ambulances while on way to hospitals, patients under monitoring at homes or offices, patients both in critical and routine care. The system consumes the high dimensional and high volume data to detect any indication of attack on the integrity of data flowing from IoT domain into the cloud domain and among the clouds. An efficient system can weed out attacks fast and with high accuracy so as to let the medical staff provide the best care without unduly worrying about false alarms. As the dataflows pass through the edge cloud gateways, the extracted meta-information is passed though the SAE executing in edge processors. In the case of the normal traffic flow, the meta-information is reconstructed with RMSE below the preset threshold. In cases where the traffic consisting of patient data has been intruded upon, the meta-information about the traffic flow is affected. In this case, the meta-information can no longer be reconstructed with low RMSE, and the system indicates intrusion.

The edge cloud models are trained on examples collected from their respective coverage area. The core cloud model is constituted from the trained layers of edge clouds. The model is further trained using the data from its own area not overlapping edge cloud areas. Fig. 10 shows the system workflows for initial training as well as during normal operation.

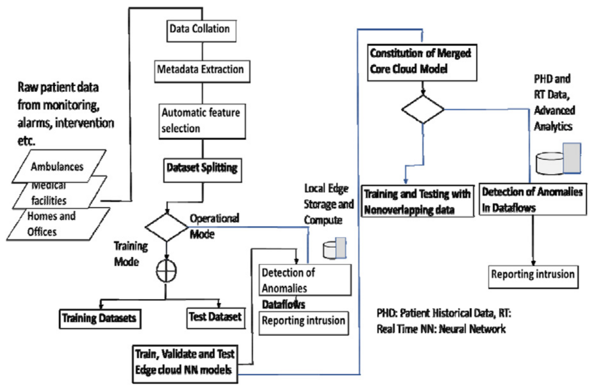

Data Collation
Metadata Extraction
Automatic feature selection
Dataset Splitting
Raw patient data from monitoring, alarms, intervention etc.
Ambulances
Medical facilities
Homes and Offices
Operational Mode
Training Mode
Training Datasets
Test Dataset
Train, Validate and Test Edge cloud NN models
Local Edge Storage and Compute
Detection of Anomalies Dataflows
Reporting intrusion
Constitution of Merged Core Cloud Model
PHD and RT Data, Advanced Analytics
Training and Testing with Nonoverlapping data
Detection of Anomalies In Dataflows
Reporting intrusion
PHD: Patient Historical Data, RT: Real Time NN: Neural Network

Fig. 10. Training and operational workflows.

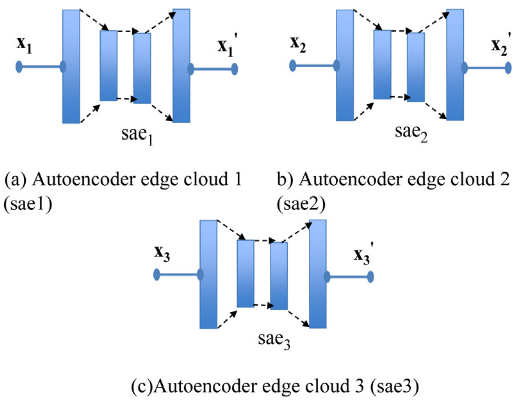

x1
x1'
sae1
(a) Autoencoder edge cloud 1 (sae1)
x2
x2'
sae2
b) Autoencoder edge cloud 2 (sae2)
x3
x3'
sae3
(c)Autoencoder edge cloud 3 (sae3)

Fig. 11. Three edge cloud autoencoders.

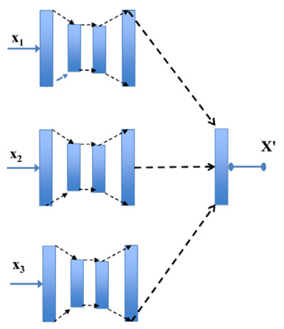


Fig. 12. A merged model with all edge layers reused.

Algorithm 1 Edge cloud model creation and training

```
Initialization and Input definition:
X ← dataset
Split X into $X_{train}$ (80%) and $X_{test}$ (20%)
Extract input dimension from shape of $X_{train}$
Number of edge clouds = $n_e$
Number of layers in each edge cloud = $l_e$
Encoding dimension =m
Create edge cloud models
for edge_cloud in range 1 to $n_e$
    define input layer
    for layer in range h=1 to $l_e$
        if h<$l_e$/2
            define encoding layer h
            set activation function ← 'relu'
            set regularization factor ← $10*e^{-8}$
            set sparsity factor
        elseif
            define decoding layer h
            set activation function ← 'relu'
        endif
    endfor
endfor
Configure model
for sparse autoencoder m = 1 to $n_e$
        create autoencoder with input layer m and output layer m
        compile the model
        set optimization ← 'Adam'
        set loss ← 'mean-squared-error'
        set metric ← 'accuracy'
endfor
Train model
for sparse autoencoder m = 1 to $n_e$
    specify test data
    set epochs ← 100
    set batch_size ← 64
    specify validation data
endfor
```

Algorithm 1 gives an idea of creation, configuration and training of edge clouds [51]. The parameters used are indicative and various setting were tried for good outcome. The code has been implemented in Keras and TensorFlow with three edge-cloud models as shown in Fig. 11.

The aggregator in the simple mode merges all the layers of the edge models and produces a composite model (Figs. 9 and 12). Algorithm 2 gives a simple implementation of this operation. The merged model has layers taken from the edge clouds. The aggregator can take all or merge layers selectively to improve the outcome.

Algorithm 2 Core cloud model creation and training

```
    …
Initialization and input definition
…
Creation of merged core model
for edge clouds 1 to m
    select input layer
    select output layers
endfor
define model output from merged output layers
create merged models from selected input and output layers
Configure merged model
…
Train the merged model
…
```

Fig. 13 shows merging of layers from different edge clouds to form the core cloud. We shall compare the training times of core cloud neural network models created by reusing trained layers from edge cloud neural networks with those that do not use trained layers.

5.6. Computational complexity of merged model

In assessing complexity of neural network models, it helps to make some simplifying assumptions. We assume the same number of neurons in each layer and the number of layers equal to number of neurons.

In forward propagation training step, we have matrix multiplications and computation of activation function. The number of matrix multiplications $n_{mul} = n_{layers}.\ n^3$ in other words n_{mul} has the asymptotic run time of $O(n^4)$. The activation function is evaluated elementwise which gives a complexity of $O(n).O(n) = O(n^2)$. The total forward propagation runtime becomes $O(n^4 + n^2)$ which is of the order of $O(n^4)$.

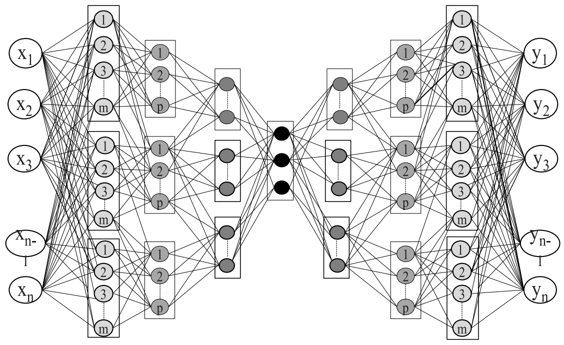


Fig. 13. A merged model with selective layer reuse.

Each epoch also involves back propagation through the n layers. For calculation of error, we have O(time $_{error}$) = (n^4) and for calculating weights we have O(time $_{weights}$) = O(time $_{error}$) + n^3. Now taking n gradient descent calculations we have overall complexity as O(n^5) [57,58]. This is polynomial complexity that will lead to long training convergence times. It turns out that back propagation is much slower than forward propagation. This raises the question whether the complexity of the model can be reduced to limit the training time at the cost of only a non-substantial performance loss. The answer is 'yes'.

The way our method reduces the computational complexity is by reducing the number of trainable parameters. The baseline model with 8 fresh layers i.e., not exposed to any training example consists of 31 521 trainable parameters. Our merged model with 12 layers consists of 24 737 trainable parameters reduced by used of trained layers from the edge cloud models. We run these for 200 epochs which with 8000 training examples and a batch size of 128 comes to about 12 500 steps. This reduction of parameters though merged model reduces the training time drastically. We see these results in Section 6.4.

6. Evaluation and results

In this section, we first discuss the test datasets, two of them publicly available and the other that has been generated by the authors. Then, we use these datasets to evaluate the effectiveness of the proposed model and draw conclusions.

6.1. UNSW-BOT-IoT and UNSW15 dataset

The BOT-IoT dataset has been made available by the UNSW Canberra Cyber Center and updated in November 2018. It was created using a realistic network environment with simulated existence of IoT devices in the virtual network. Commonly used older sets like KDDCup1990 and NSL-KDD lack IoT generated traffic and do not take care of new types of attacks. In the selected dataset, the testbed consists of three parts: network platforms, simulation of IoT services and extraction of features and forensic analysis. Normal background traffic is constantly generated and attacks were interspersed. The environment incorporates a combination of normal and botnet traffic. The dataset includes five botnet attack types — DDoS, DoS, Service Scan, Keylogging, and Data exfiltration attacks. A total of 48 features have been included. UNSW15 from the same organization has a total of 48 basic features. It has nine attack types — DDoS, DOS, Service Scan, Keylogging and Data Exfiltration A part of the dataset (varying from 6,000 to 10,000 records out of millions of available examples) has been used in various runs with 80:20 ratio of training and test examples. Table 2 gives some sample features of the BoT_IoT dataset. The full dataset can be seen in [59,60].

Table 2
The feature set of the BOT-IoT dataset.

Feature	Description	Feature	Description
saddr	Source IP address	spkts	Source-to destination packet count
daddr	Destination IP address	pkts	Destination-to-source packet count
pkts	Total transaction packet count	srate	Source-to-destination packets per second
bytes	Total transaction byte count	N_IN_Con n_P_SrcIP	Number of inbound connections per source IP address.
dur	Total duration of the record	min	Minimum duration of aggregated records
Proto	protocols present in network	max	Maximum duration of aggregated records
Rate	Total packets per second in transaction	attack	Class label: 0 for Normal 1 for Attack traffic
Flags number	Numerical representation of feature flags	State number	Numerical representation of feature state

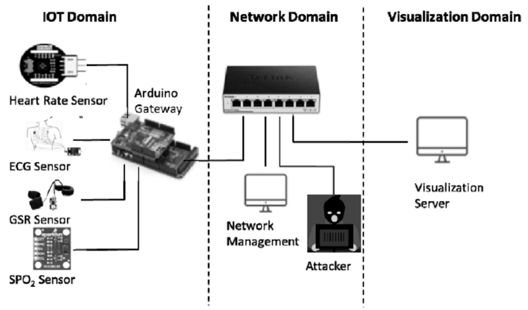


Fig. 14. Healthcare testbed for dataset generation.

6.2. Data generated on the testbed

The BOT-IoT dataset consists of meta-information extracted from the flows from many IoT devices. While this will be indistinguishable from that generated in the healthcare systems, we created a healthcare-specific dataset in the testbed at Washington University in St. Louis. The testbed set-up shown in Fig. 14 has been adapted from [43].

(a) The IoT Domain: Consists of a set of health IoT sensors and an Arduino Mega based microcontroller as the gateway for health IoT sensors. Because of the absence of any Wi-Fi or Ethernet port on this microcontroller, an external Ethernet shield was attached. The following sensors have been used: galvanic skin response, which measures skin tightening due to stress, pulse oximeter for measuring the oxygen level in the blood, and body temperature sensor.

(b) The Network Domain: An Ethernet switch to which the microcontroller and three servers were connected, all configured as one private network with each device assigned a private IP. One of the servers was used to mirror and record traffic coming to the network from the IoT domain. The computer through which attacks were generated served as an insider attacker.

(c) The Visualization Domain: A server with Ubuntu Linux operating system was used to visualize the patient's data generated through the sensors described above. The sensor values and the meta-information could be visualized and saved in files for processing.

(d) Attacker: A server with the Kali Linux operating system was used to cause malicious activities like sniffing and tampering the dataflows.

Table 3
Features in the IoT testbed dataset.

Feature	Description	Feature	Description	Feature	Description
SrcAddr	Source Address	DIntPkt	Destination inter packet arrival time (ms)	Load	Bits per second
DstAddr	Destination Address	SIntDist	Source inter packet arrival time distribution	Loss	Packets transmitted or dropped
Sport	Source Port #	DIntDist	Destination inter packet arrival time distribution	sMinPktSz	Min packet size for source traffic
Dport	Destination Port #	SIntPktAct	Source active inter packet arrival time	dMinPktSz	Max packet size for source traffic
SrcBytes	Source-to-destination byte count	DIntPktAct	Destination active inter packet arrival time	pLoss	Percent packet transmitted or dropped
DstBytes	Destination-to-source byte count	SrcJitter	Source jitter (ms)	pSrcLoss	Percent source packet transmitted or dropped
SAppBytes	Source to destination application bytes	DstJitter	Destination jitter (ms)	pDstLoss	Percent destination packet transmitted or dropped
DAppBytes	Destination to source application bytes	sMaxPktSz	Max packet size for source traffic	Dur	Duration of a flow
SrcLoad	Source bits/s	dMaxPktSz	Max packet size for dest. traffic	Trans	Aggregation record count
DstLoad	Destination bits/s	DstGap	Destination bytes missing	TotPkts	Total transaction packet count
SrcGap	Source bytes missing	SIntPkt	Source interpacket arrival time (ms)	TotBytes	Total transaction byte count

The normal and attack datasets have the features given in Table 3.

6.3. Implementation platforms and software tools used

The edge and core models have been tested on a variety of hardware using several software tools and datasets. The code implementing the models has been developed in Python on the Anaconda/Spider platform. Some parts of the codes were ported to MATLAB and tested using the built in deep learning library to verify the results. The hierarchical merged model with three edge clouds and the core cloud were run on a Mac with 8 core CPU as well as a Windows machine with GPU Nvidia GTX 1080. During the review process the models were also trained and tested on Google Colab cloud platform using TensorFlow version 2.x and Keras. The Colab platform offers a variety of CPUs and GPUs including Nvidia K80s, T4s, P4s and P100s. The testbed described in Section 6.2 was used to generate normal and attack data. Normal data emanating from sensors in the IoT domain were recorded over a number of sessions with different volunteers. The data is anonymized with no means to trace back to the individuals involved. Attacks were simulated using a Kali driven system. In the present evaluation, the meta-information is extracted using Argus Network Management System [40] and ranked using the Weka machine-learning tool [41]. In the future work, automatic extraction will be integrated with the system.

6.4. Results

The results discussed in this section are for the training of neural networks at the edge cloud and a comparison of training of the core cloud neural network with and without layer reuse and efficacy of the system in filtering out attacks.

(a) Training and Testing of SSAE at the Edge Clouds

The configuration discussed in Section 5 and represented in Figs. 5 and 6 was used as the basis for training and testing. The generated dataset was randomized, and mutually exclusive parts were selected to train the AEs in the three edge clouds. All the examples in the training datasets constituted normal traffic. Some parts of each of the segregated datasets were kept apart to be used as test datasets. The models did not see these datasets in the training phase. Some examples, with known attacks, were separated from the training and test datasets for evaluation of detection performance.

The training and testing results are shown in Fig. 15a to c. It can be seen from the figures that the training accuracies are excellent, and the model generalizes well. The blue lines represent training losses, while the green lines represent test losses. Each epoch represents a combination of one forward and one back-propagation iteration of the complete dataset through the SAE. As the number of epochs increases, the model gets trained, and the losses come down. The losses of the models on the test data settle down to their low value close to the training losses in 40–60 epochs of training.

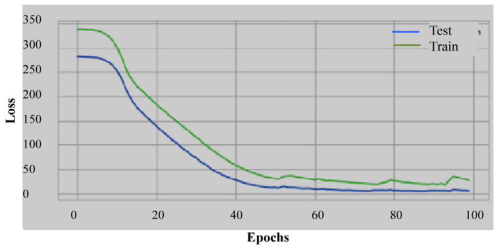
(b) Training and testing of SAE at the core cloud.

The training and testing results are shown in Fig. 16(a) for the merged model in the core cloud. Fig. 16(b) shows results for the merged model with cross-training. In the latter, the cloud data are swapped among themselves.

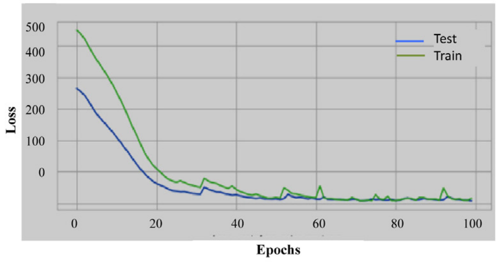
It is seen from Fig. 16a and b, that in both cases, the model takes a substantially smaller number of iterations and, consequently, less time, compared to edge cloud training. It is also seen that the cross-trained edge models result in somewhat lower training time for the core cloud model (the result stabilizes in less than four epochs against five epochs). To compare the edge cloud and the core cloud training times, we present Figs. 17 and 18. Fig. 17 shows the training and validation losses and the gap between the two, at the edge cloud, with an increasing number of epochs. It is seen that the performance stabilizes between 35 and 40 epochs for the edge clouds for the selected dataset. Fig. 18 shows the training speed of the merged model in the core cloud. It is seen that because of the use of already trained layers and from the edge clouds and the reduction of trainable parameters at the core cloud, the model in the core cloud stabilizes between 6 and 8 epochs. This is a significant improvement in the speed of training of the model in the core cloud because of the reuse of layers trained in the edge clouds.

In most runs with different data, it is seen from the test results that the model generalizes well. The test losses were initially higher than the training losses but after a few iterations they settle down close to train losses. Test accuracies are high and close to those obtained during testing both for edge and core clouds. Occasionally overfitting happens which manifests as better test performance compared to train performance. This has been effectively controlled by using appropriate regularization and by not over-training the cloud beyond convergence after 5 or 6 epochs.

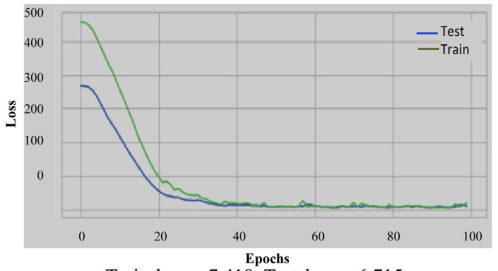
It would be apt to discuss the hyper-parameter settings briefly for the models. We use Adam optimizer, which provides stochastic gradient-based optimization. It computes adaptive learning



Train-loss = 4.697, Test-loss = 17.595,
Train-accuracy = 0.993, Test-accuracy = 0.995
(a) Edge cloud 1

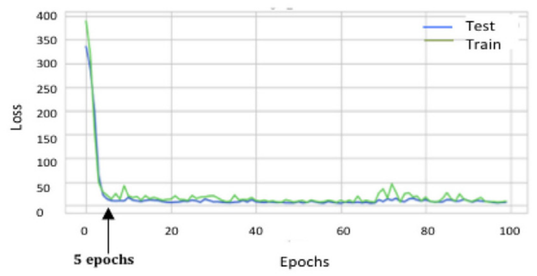


Train-loss = 8.526, Test-loss = 8.661,
Train-accuracy = 0.986, Test-accuracy = 0.985
(b) Edge cloud 2

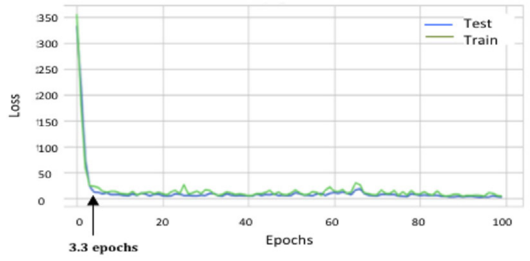


Train-loss = 7.418, Test-loss = 6.715,
Train-accuracy = 0.993, Test-accuracy = 0.992
(c) Edge cloud 3

Fig. 15. Training and test performance of edge clouds.



Train-loss = 4.517, Test-loss = 6.151 Train-accuracy = 0.989, Test-accuracy = 0.990
(a) Core cloud merged model



Train-loss = 2.1575, Test-loss = 4.1718,
Train-accuracy = 0.992, Test-accuracy = 0.993
(b) Core cloud merged model (cross-trained)

Fig. 16. Training and testing results.

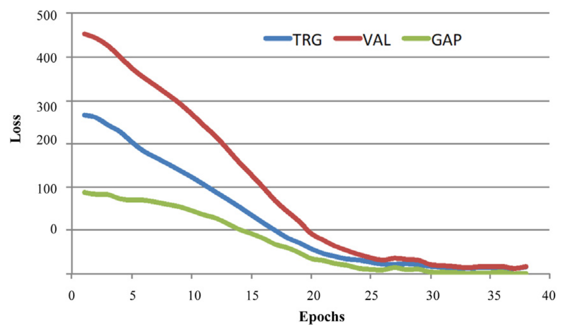


Fig. 17. Edge cloud model training.

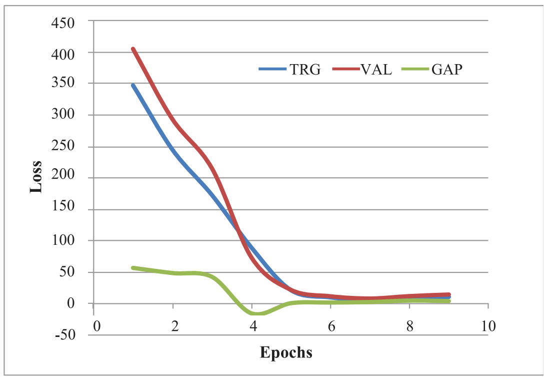


Fig. 18. Core cloud model training.

rates for different parameters from estimates of first and second moments. The exponential decay rates ($ß_1$ and $ß_2$) for the moments go as hyperparameters. We choose learning rate = 0.001, $ß_1$ = 0.8 and $ß_2$ = 0.9 and 0.999 according to the guidelines in [47]. The datasets were divided into test and train datasets in the ratio 80:20. The loss function used was the mean square error. The activation functions used were 'tanh' or 'RELU,' chosen for the best performance. The batch size for training was taken as 100. Several experiments were conducted to decide on the most appropriate number of hidden layers and the number of neurons per layer for each edge cloud and the core cloud.

(c) *Training time analysis*

In Section 5 we saw a naïve analysis of computational complexity of neural network model. We saw that the overall complexity with forward and backward is of the order of O(n^5). This denotes high complexity which goes up very fast with the number of layers, number of neurons and the training examples. For the kind of SAEs that we discussed, the theoretical training time with greedy layer wise training becomes impractically high for real time systems. From the related works we can see that researchers have suggested innovative ways to improve the accuracy of prediction but very few have written about the training time of their models. One of the few papers that discuss training times of their models is [61]. The author performed baseline

Table 4
Comparison of unmerged and merged models.

	8 layer unmerged model	12 layer merged model
Layers	180,90,60,30,30,60,90,180	60 × 3[a], 60 × 3[a], 30 × 3[a], 90 × 2
Time for 100 epochs	1 h 24 min and 30 s	1 h and 2 min

[a]Trained layers from edge clouds.

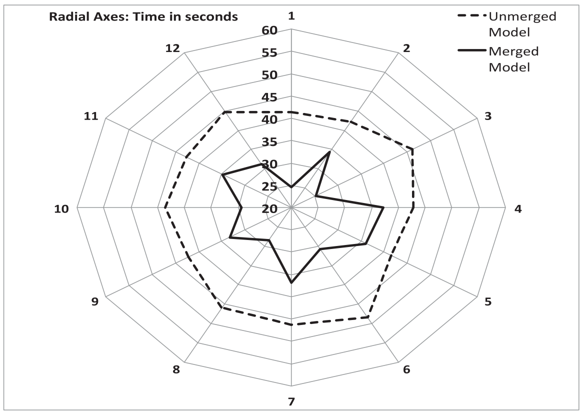


Fig. 19. Radar chart comparing training times of merged and unmerged models.

Table 5
Confusion matrices for attack detection.

Sl. No.	Attack as Attack (TP)	Attack as Normal (FN)	Normal as Attack (FP)	Normal as Normal (TN)	Total vectors	Accuracy (%)
1.	200	0	19	355	574	96.69
2.	188	0	0	511	699	100.00
4.	96	0	3	232	331	99.09
5.	92	16	0	184	292	94.52
6.	100	0	0	280	380	100.00
7.	80	17	0	243	340	95.00
8.	110	1	1	287	399	99.50
9.	27	1	1	100	129	98.45
10.	126	12	0	398	537	97.76

Table 6
Attack detection performance.

		Actual		
		Positive	Negative	Total
Predicted	Positive	56	2	58
	Negative	1	218	219
	Total	57	220	277
	TPR, Recall	0.9859		
	FPR	0.0064		
	Accuracy	0.9920		
	Precision	0.9655		
	F1-Score	0.9755		

experiment with 20 epochs of pre-training of RBM followed by 10 epochs of fine-tuning by backpropagation. The depth of the model is 5 layers with the biggest dimension as 1000 and activation function is sigmoid. The training and test datasets have 60,000 and 10,000 examples respectively. Layer wise training for 20 epochs takes 3 h, 14 min and 43 s while backward propagation takes 2 h, 16 min and 59 s a total of 5 h 31 min and 42 s. With their synchronized pre-training the total time taken is 4 h 4 min and 53 s.

In our case the edge cloud models are trained in a greedy layer wise manner take about 125 s for 4000 training examples and 1000 test examples on M400M Quadro GPU on an Intel Xeon v5 CPU. The depth of these models is kept at 4. These models take 35–40 epochs to trains. The unmerged core model has depth of 8 and takes number of epochs of the same order to train without the merged model technique that we have proposed. By utilizing the trained layers from the edge cloud, we reduce the number of epochs to 5–6. The epochs are longer because of the native complexity of the model but the model still takes less time than with unmerged model. The baseline 8-layer model with layer configuration (180,90,60,30,30,60,90,180) and 31 521 trainable parameters takes on an average 176 s for 7000 training examples and 1500 test examples run for 100 epochs. The merged model much larger with 12 layers, partly constructed by using trained layers from the edge clouds. The trainable parameters were reduced to 27,275, After several runs, we see that the merged 12-layer model takes less time than even a much smaller 8-layer unmerged model. Table 4 summarizes these results.

A comparison of training in both cases is given in Fig. 19. The radar chart in Fig. 19 clearly brings out the reduction in training time of merged model as compared to the unmerged model. Through several run it is seen that the merged models give on an average 26.2% reduction in training times.

(d) Performance of the MUSE system in filtering out the attack cases

To assess the intrusion detection performance of the MUSE system, we used the attack both from the UNSW datasets as well as our testbed. We used attack cases that potentially alter the metadata information of the dataflows. Specifically, fuzzers that break applications by injecting random data, backdoor that may grant remote access and cause abnormal traffic to flow, DOS attacks that affect access to network resources, reconnaissance traffic to network devices and end-devices. The test data thus consists of carefully chosen attack and normal records. We run these through the trained models and enumerate results. It is seen that the anomalous data produce very high RMSE, and it is easy to fix a threshold value that decides whether the data has been affected by malicious activity. The confusion matrix for 10 out of several runs is given in Table 5. These random runs give a low average false positives rate of 0.5% and unseen attack detection accuracy generally in the range 95% to 100%. This is a marked improvement when compared with the previously unseen attack detection accuracy of 92%–93% achieved with the unmerged model.

The records taken in the test dataset for the attacks mentioned above were rotated from the large UNSW datasets. Multiple runs were made with the threshold that gives the best performance. In our case a threshold of MSE of 10 was this demarcation point. Table 6 gives various figures for the average of many runs with this threshold. The high accuracy could refer to high true positives or true negatives i.e., attacks detected as attacks and normal traffic detected as normal. Since in our case of normal traffic being labeled at attack triggers investigation, which may be expensive, precision provides a good performance measure. Precision being high at 96.5% assures that the model is good at detecting actual attack incidences as attacks. Since the cost of false negatives is also high in our case we can have comfort in high recall (or true positive rate) which is 98.6%. The false positive rate (FPR) is 0.64%. An F1 score of 97.5% indicates both low false positives and false negatives are low and recall and precision are well balanced.

7. Comparison with other works

Comparative results have been given in Table 7. Rows 1 to 3 are from works reported in the years 2020 and 2021. The authors have proposed modified versions of the standard deep learning models like CNN, LSTM and DBN. In [24] the authors applied

Table 7
Comparison with other works.

	Reference	Model, Dataset	Accuracy	Timing analysis	Remarks
1	[24]	CNN, CICIDS2017	96.55%	–	Cloud computing environment
2	[25]	BMO-DBN	97.65	–	Pure DBN gives 96.17%
3	[26]	CNN, LSTM, generated dataset	Training: CNN 96.7 LSTM 98.5 Test 90.2, 95.1	–	Edge cloud environment for automatic vehicles
4	[49]	SVM NSL-KDD	NSL-KDD8 88.32% UNSW 93.3%	–	Baseline, shallow model
5	[49]	RBM with KDDcup 1999, UNSW15, NSL-KDD	KDDCup99: 90.99%, UNSW-NB15, 95.84%, NSL-KDD 97.11%	–	
6	[27]	Modified LeNet-5	NSL-KDD 87.30% KFFCup99 81.94%		NSL-KDD/KDD Cup Detection rate=93.86/99.82 False Positive Rate=21.38/98.65
7	[28]	RBC-IDS	KDDCup99 90%-99.91%	Training 31.5 s, Test 1.62 s	WSN cluster with 20 sensors, up to 3 hidden layers
8	[28]	ASCH-IDS	KDDCup99 Upto 99.83%	Training 17.1 s Test 0.86 s	
9	[29]	BLSTM-RNN	UNSW15 95%		
10	[62]	H2O deep learning.	NSL-KDD, Train 99.5%, Test 83%		
11	[30]	NDAE SAE	KDD C 97.9%, NSL-KDD 97.85%		
12	[31]	STL-IDS, NSL-KDD	NSL-KDD 84.96%		Binary classification
13	[48]	DNN UNSW	65%–75% (Train)		
14	[48]	DNN 5-layer UNSW-NB15 binary classification	76.3%		
15	This work	MUSE UNSW-15	Test 95–99.5%	27% speedup over unmerged	Multi-cloud, AI for attack detection

First stage trained with UNSW-NB15 with both normal and attack examples.

CNN to cloud computing and trained with CICIDS2017 dataset to achieve 96.55% accuracy. In [25] the authors present a data offloading mechanism with cyber-attack detection (DLTPDO-CD) involving barnacles mating optimizer (BMO-DBN0 to get accuracy of 97.65%. In [26] authors propose an improved LSTM model to achieve 98.5% training and 95.1% test accuracy. Rows 4 and 5 show shallow (baseline) and deep learning results from the work in [49]. Authors get improvement over the baseline SVM and best accuracy for NSL-KDD of 97.11%. In row 6 we summarize the results achieved by authors in [27]. The authors have worked with modified LeNet-5 model to achieve an overall accuracy of 97.53% with KDDCup99 dataset. Row 7 and 8 contains results of the work in [28] in which the authors propose Restricted Boltzmann Machine-based clustered IDS(RBC-IDS) in WSN environment using KDDCup99. In row 9 the authors in [29] use BLSTM RNN and achieve 95% accuracy with UNSW15 dataset and higher with older datasets. In the work at row 10 the authors use H2O deep learning based binomial and multinomial models with NSL-KDD to achieve 99.5% on training and 83% on test dataset [62]. In the work at row 11 the authors use NDAE SAE with KDDCup99 and NSL-KDD get accuracy of 97.9% and 97.85% [30]. In [31] (row 12) authors propose AE based STL-IDS and test it with NSL KDD dataset to obtain 84.96% accuracy. In rows 13 and 14, the authors use deep neural network (DNN) to achieve 65%–75% training accuracy with the UNSW-NB15 datasets and binary classification accuracy of 76.3% [48]. This is markedly lower than 95%–99% accuracy achieved with the KDDCup'99 datasets.

The last row indicates the accuracy obtained by MUSE trained with UNSW-NB15 datasets. These figures obtained over many experiments show comparable or better results than the compared works.

8. Conclusions

From the discussion in this paper we see that some researchers have reported use of deep learning in the form of artificial neural network models for detection of attacks in different environments [30]. Very few of these works are in the cloud environment and hardly any in multi-cloud environment. To the best of our knowledge, no investigation is available for the IoT-multi-cloud infrastructure, especially with hierarchical and merged AEs with layer reuse. The MUSE system proposed in this paper fills this gap. The proposed hierarchical model works very well with the hierarchical structure of the healthcare network. The distributed nature of the intrusion detection system has models of increasing complexity from IoT to the core clouds. This makes the implementations commensurate with the processing capabilities available at different levels. One common problem with sophisticated deep learning models is their time complexity. We have explored reducing the training time at the core clouds through innovative reuse of trained layer from the edge clouds. The method works by reducing the number of parameters to be trained in the core cloud. We are able to achieve fast training rates for sizeable neural network models at the core clouds by aggregating the trained layers of the edge cloud neural network models. Exceeding our expectations, not only the timings improve drastically, the accuracies are much better than those that are achieved by training new SAE at the core clouds and also compared to many of the cited works. It is seen that the training time of a 12-layer merged model that reuses trained layers from the edge cloud models takes 10.1% to 29.2% less time than a smaller 8-layer unmerged model that does not reuse the training carried out at the edge clouds. Accuracies of 92%–93% were obtained with the unmerged models, while the merged models achieved accuracies of more than 95% and more often in the range of 98%-99.6%. In real life, merged models trained on a combination of historical patient data stored in the core clouds along with more recent data patient data from the IoT gateways and the edge clouds, are expected to give results comparable to those that we have been obtained under simulated settings.

Healthcare systems are different from many others as human life and well-being is involved. Decisions given by the deep learning system should stand to scrutiny and explanation. Medical experts should be able to question the system about the reasons for giving a certain diagnosis or prognosis. Patients should be able to ask why the doctor has prescribed a particular line of treatment. This will generate confidence and make these systems acceptable. We plan to carry forward our research in this direction.

Nomenclature

Acronyms /Symbols	Expansion
b	The bias term
J	The Cost function
L	The Loss function
T	Transposition operation
w	Weights
x_i	The ith input
y	Output
β	Coefficient for Ω_{sparsity}
λ	Coefficient for Ω_W
Ω_W	Weight regularizer
$\Omega_{sparsity}$	Sparsity regularizer
AE	Autoencoder
APT	Advanced Persistent Threat
CIA	Confidentiality, Integrity and Availability
CICIDS2017	Intrusion Detection Evaluation Dataset
CNN	Convolutional Neural Network
DBN	Deep Belief Network
DDoS	Distributed Denial of Service
DFD	Dataflow Diagram
DNN	Deep Neural Network
DRL	Deep Reinforcement Learning
DT	Decision Trees
ECG	Electrocardiogram
FFN	Feedforward Neural Network
FPR	False Positive Rate
GDP	Gross Domestic Product
GPU	Graphic Processing Unit
HIDS	Host-based Intrusion Detection System
IoT	Internet of Things
ISP	Internet Service Provider
KNN	K-Nearest Neighbors
LSTM	Long Short-Term Memory
MSE	Mean Square error
MUSE	Merged Hierarchical Deep Learning System with Layer Reuse for Security
NIDS	Network-based Intrusion Detection System
OT	Operational Technology
RBM	Restricted Boltzmann Machine
RELU	Rectified Linear Unit
RF	Random Forest
RNN	Recurrent Neural Network
SIM	Subscriber Identification Module
SSAE	Sparse Stacked Autoencoder
SVM	Support Vector Machine
UI	User Interface
UNSW	University of New South Wales
VNS	Virtual Network Service
WSN	Wireless Sensor Network

CRediT authorship contribution statement

Lav Gupta: Conceptualization, Methodology, Investigation, Software, Formal analysis, Original draft, Revisions. **Tara Salman:** Resources, Review and editing. **Ali Ghubaish:** Review and editing, Validation, Resources. **Devrim Unal:** Visualization, Writing, Review, Funding acquisition. **Abdulla Khalid Al-Ali:** Visualization, Writing, Review, Project administration, Funding acquisition. **Raj Jain:** Visualization, Conceptualization, Supervision, Review, Project administration, Funding acquisition.

Declaration of competing interest

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

Funding

This publication was made possible by NPRP grant 10−0125−170250 from the Qatar National Research Fund (a member of Qatar Foundation) and NSF grant CNS-1718929 from the National Science Foundation. The findings achieved herein are solely the responsibility of the author[s].

References

[1] Centers for Medicare and Medicaid Services, National Health Expenditure Data- NHE Fact Sheet, CMS.gov, 2021, last updated 15th Dec 2021. Last accessed 1st 2021.
[2] T.L. Rodziewicz, B. Houseman, J.E. Hipskind, Medical Error Reducton and Preventon, StatPearls, Treasure Island (FL), StatPearls Publishing, Jan 2022.
[3] IDC expects 2021 to be the year of multi-cloud as global COVID-19 pandemic reaffirms critical need for business agility, 2021, https://www.idc.com/getdoc.jsp?containerId=prMETA46165020 last updated March 2020, last accessed 1st July 2021.
[4] Healthcare and cloud adoption in 2021, OpsCompass, 2021, https://opscompass.com/wp-content/uploads/2021-oc-whitepaper-healthcare-multicloud-03122021.pdf, last accessed 1st 2021.
[5] H. Aziz, A. Guled, Cloud computing and healthcare services, J. Biosensors Bioelectron. 7 (3) (2016) Medical IoT Technology in US Hospitals Helps to Reduce Costs and Improve Care, S & P Global Market Intelligence, Brian O' Rourke, 2020 https://www.spglobal.com/marketintelligence/en/news-insights/blog/medical-iot-technology-in-us-hospitals-helps-to-reduce-costs-and-improve-care, last accessed 14th July, 2021.
[6] Price Waterhouse Cooper, The global state of information security survey 2018, 2018, Available: www.pwc.com/gsiss, last accessed 14th 2021.
[7] P. Potter, The current state of security and risk in healthcare, 2020, Available at https://imaginenext.ingrammicro.com/b2b-tech-talk/the-current-state-of-security-and-risk-in-healthcare, last accessed 29th July, 2021.
[8] Data Breach Investigations Report, DBIR 2021 verizon, 2021, Available at https://www.verizon.com/business/resources/reports/dbir/, last accessed 29th 2021.
[9] Cyber Claims Study, Net diligence, Whitepaper, 2020, Available at: https://netdiligence.com/wp-content/uploads/2021/03/NetD_2020_Claims_Study_1.2.pdf, last accessed 1st 2021.
[10] S.J. Choi, M.E. Johnson, Do hospital data breaches reduce patient care quality, 2019, arXiv:1904.02058 [econ.GN].
[11] Medical Device Recall, FDA, 2020, Available at: https://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfRES/res.cfm, last accessed 5th 2021.
[12] D. McDonalds, MythBusters: Healthcare Cloud Management, Doug McDonalds, 2021, Extreme Networks.
[13] K.A. Makdi, F. Sheldon, A.A. Hussein, Trusted security model for IDS using deep learning, in: 3rd International Conference on Signal Processing and Information Security (ICSPIS), 2020.
[14] D.E. Kim, M. Gofman, Comparison of shallow and deep neural networks for network intrusion detection, in: IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, 2018, pp. 204–208.
[15] Y. Xin, et al., Machine learning and deep learning methods for cybersecurity, IEEE Access 6 (2018) 35365–35381.
[16] F. Farahnakian, J. Heikkonen, A deep auto-encoder based approach for intrusion detection system, in: 20th International Conference on Advanced Communication Technology (ICACT), Chuncheon-si Gangwon-do, Korea (South), 2018, p. 1.
[17] K.K. Nguyen, D.T. Hoang, D. Niyato, P. Wang, D. Nguyen, E. Dutkiewicz, Cyberattack detection in mobile cloud computing: a deep learning approach, in: 2018 IEEE Wireless Communications and Networking Conference (WCNC), Barcelona, 2018, pp. 1–6.
[18] Ali Bou Nassif, Manar Abu Talib, Qassim Nassir, Halah Albadani, Fatima Dak Albab, Machine learning for cloud security: A systematic review, IEEE Access (2021).
[19] V. Hayyolalam, M. Aloqaily, O. Ozkasap, M. Guizani, Edge intelligence for empowering IoT-based healthcare systems, 2021, arXiv:2103.12144 [cs.LG].
[20] H. Elayan, M. Aloqaily, M. Guizani, Digital twin for intelligent context-aware IoT healthcare systems, IEEE Internet Things J. 202.

[21] F. Zaman, M. Aloqaily, F. Sallabi, K. Shuaib, J.B. Othman, Application of graph theory in IoT for optimization of connected healthcare system, in: GLOBECOM 2020-2020 IEEE Global Communications Conference, 2020, pp. 1–6.

[22] M. Al-Khafajiy, S. Otoum, T. Baker, M. Asim, Z. Maamar, M. Aloqaily, M. Taylor, M. Randles, Intelligent control and security of fog resources in healthcare systems via a cognitive fog model, ACM Trans. Internet Technol. (2021).

[23] S. Hizal, Ü. Çavuşoğlu, D. Akgün, A new deep learning based intrusion detection system for cloud security, in: 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications, 2021.

[24] L. Chen, X. Kuang, A. Xu, Siliang Suo, Y. Yang, A novel a novel network intrusion detection system based on CNN, in: Eighth International Conference on Advanced Cloud and Big Data (CBD), 2020.

[25] T. Gopalakrishnan, et al., Deep learning enabled data offloading with cyber attack detection model in mobile edge computing systems, IEEE Access (2020).

[26] Y. Xun, J. Qin, J. Liu, Deep learning enhanced driving behavior evaluation based on vehicle-edge-cloud architecture, IEEE Trans. Veh. Technol. (2021).

[27] W. Lin, H. Lin, P. Wang, B. Wu, J. Tsai, Using convolutional neural networks to network intrusion detection for cyber threats, in: 2018 IEEE International Conference on Applied System Invention (ICASI), Chiba, 2018, pp. 1107–1110.

[28] S. Otoum, B. Kantarci, H.T. Mouftah, On the feasibility of deep learning in sensor network intrusion detection, IEEE Netw. Lett. (2019).

[29] B. Roy, H. Cheung, A deep learning approach for intrusion detection in internet of things using bi-directional long short-term memory recurrent neural network, in: 28th International Telecommunication Networks and Applications Conference (ITNAC), Sydney, Australia, 2018, pp. 1–6.

[30] N. Shone, T.N. Ngoc, V.D. Phai, Q. Shi, A deep learning approach to network intrusion detection, IEEE Trans. Emerg. Top. Comput. Intell. 2 (1) (2018) 41–50.

[31] R. Vinayakumar, et al., Deep learning approach for intelligent intrusion detection system, IEEE Access (2019) 41525–41550.

[32] Adel Abusitta, Martine Bellaiche, Michel Dagenais, Talal Halabi, A deep learning approach for proactive multi-cloud cooperative intrusion detection system, Future Gener. Comput. Syst. 98 (2019) 308–318.

[33] Mingshu He, Xiaojuan Wang, Junhua Zhou, Yuanyuan Xi, Lei Jin, Xinlei Wang, Deep-feature-based autoencoder network for few-shot malicious traffic detection, Secur. Commun. Netw. (2021).

[34] UFLDL tutorial autoencoders, 2021, Available at : http://ufldl.stanford.edu/tutorial/unsupervised/Autoencoders/, last accessed on 16th 2021.

[35] L. Gupta, Hierarchical deep learning for cybersecurity of critical service systems, accepted for presentation in IEEE world conference on smart trends in systems, Secur. Sustain. (2018).

[36] W. Widanagamaachchi, Y. Livnat, P. Bremer, S. Duvall, V. Pasucci, Interactive visualization and exploration of patient progression in a hospital setting, in: AMIA Annual Symposium, 2017, pp. 1773–1782.

[37] M. Meyer, The rise of healthcare data visualization, data revolution, A J. AHIMA Blog (2017) [online] Available: https://journal.ahima.org/the-rise-of-healthcare-data-visualization/, last accessed 8th 2021.

[38] Healthcare Reference Architecture, Extreme Networks, Whitepaper, 2018, Available at, https://kapost-files-prod.s3.amazonaws.com/kapost/55ba7c9e07003d9aab000394/studio/content/5ac229fd1d425a00650000ab/revisions/1537452345-ae58e56e-49d2-4838-acb6-3ed800f0ffe1/15558-Healthcare-Reference-Architecture-2018-WP_v3.pdf, last accessed 8th 2021.

[39] J. Cusimano, Assessing the security of ICS using threat modeling, 2018, [online] Available: https://scadahacker.com/howto/howto-threatmodeling.html, last accessed 10th 2019.

[40] R. Shahan, B. Lamos, Internet of things (IoT) security architecture, 2018, [online] Available at: https://docs.microsoft.com/en-us/azure/iot-fundamentals/iot-security-architecture, last accessed on 29th 2021.

[41] N. Shevchenko, Threat modeling: 12 available methods, 2021, https://insights.sei.cmu.edu/blog/threat-modeling-12-available-methods/ CMU, 2018 Last accessed 8th 2021.

[42] K.A. McDonald, A. Wirth, The intersection of patient safety and medical device cybersecurity, HIMSS, 2018, Available at : https://365.himss.org/sites/himss365/files/365/handouts/550237082/handout-CYB4.pdf, (last accessed on 6th 2021).

[43] A. Ghubaish, Healthcare Testbed for Dataset Generation, Rotation Technical Report, Washington University in St. Louis, 2018.

[44] C. Izuakor, Managing cyber threats to operational technology, 2020, Available at https://blog.veriato.com/managing-cyber-threats-to-operational-technology, last accessed 15th 2021.

[45] Operational technology (OT) cybersecurity: 4 best practices, april, 2021, Available at https://www.beyondtrust.com/blog/entry/operational-technology-ot-cybersecurity-4-best-practices last accessed 15th 2021.

[46] M. Hassanalieragh, et al., Health monitoring and management using internet-of-things (IoT) sensing with cloud-based processing: Opportunities and challenges, in: 2015 IEEE International Conference on Services Computing, 2015, pp. 285–292.

[47] T. Bajtoš, A. Gajdoš, L. Kleinová, K. Lučivjanská, P. Sokol, Network intrusion detection with threat agent profiling, Secur. Commun. Netw. (2018).

[48] Information Technology – Security Techniques-Information Security Risk Management Standard, ISO/IEC, 2018.

[49] M. Zamani, M. Movahedi, Machine learning techniques for intrusion detection, 2019, Available at https://arxiv.org/pdf/1312.2177.pdf, last accessed 15th 2021.

[50] Sarker I.H., Deep cybersecurity: A comprehensive overview from neural network and deep learning perspective, SN Comput. Sci. (2021).

[51] M.A. Ferrag, L. Shu, H. Djallel, K.K.R. Choo, Deep, learning-based intrusion detection for distributed denial of service, attack in agriculture 4.0, Electronics (2021).

[52] X. Chen, A. Kuang, Siliang Suo Xu, Y. Yang, A novel a novel network intrusion detection system based on CNN, in: Eighth International Conference on Advanced Cloud and Big Data (CBD), 2020.

[53] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (2006) 504–507, http://dx.doi.org/10.1126/science.1127647.

[54] T. Epelbaum, Deep learning: Technical introduction deep CoRR, 2017.

[55] M. Hassanalieragh, et al., Health monitoring and management using internet-of-things (IoT) sensing with cloud-based processing: Opportunities and challenges, in: 2015 IEEE International Conference on Services Computing, New York, NY, 2015, pp. 285–292.

[56] Argus documentation, 2021, [online] Available: http://nsmwiki.org/Argus, (last accessed on 10th 2021).

[57] Computational complexity of neural networks, 2021, Available at https://kasperfred.com/series/introduction-to-neural-networks/computational-complexity-of-neural-networks, last accessed 11th 2021.

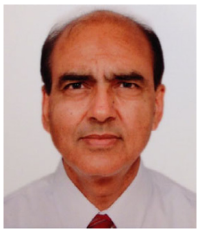
[58] V. Froese, C. Hertrich, R. Niedermeier, The computational complexity of ReLU network training parameterized, data dimensionality, 2021, arXiv:2105.08675v2 [cs.LG].

[59] N. Koroniotis, N. Moustafa, E. Sitnikova, B. Turnbull, Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset, Future Gener. Comput. Syst. (2019).

[60] N. Moustafa, J. Slay, UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set), in: IEEE Military Communications and Information Systems Conference (MilCIS), 2015.

[61] Faster learning of deep stacked autoencoders on multi-core systems using synchronized layer-wise pre-training, Anirban Santara, Debapriya Maji, DP Tejas, Pabitra Mitra and Arobinda Gupta, 2016, p. 9, arXiv:1603.02836v1 [cs.LG].

[62] S. Parampottupadam, A.N. Moldovann, Cloud-based real-time network intrusion detection using deep learning, in: International Conference on Cyber Security and Protection of Digital Services (Cyber Security), Oxford, UK, 2018, pp. 1–8.



Lav Gupta is a senior member of IEEE. He received a BS degree from the Indian Institute of Technology, Roorkee, India in 1978, and an MS degree from the Indian Institute of Technology, Kanpur, India in 1980 and a Ph.D. degree in Computer Science & Engineering from Washington University in St Louis, Missouri, USA in 2019. He is currently an Assistant Professor at the University of Missouri in St. Louis.

He has worked for about fifteen years in telecommunications planning, deployment, and regulation. He has also worked as a senior faculty of Computer Science and Access Network Planning in India and the UAE for a total of about fifteen years. He is the author of one book, 19 papers, and has been a speaker at many international seminars. His current research areas are virtual network services, multi-cloud systems, fault and performance management in cloud-based Network Function Virtualization, and application of AI in the management of virtual network services over clouds.

Tara Salman is member of IEEE. She received her BS and MS degrees from Qatar University Doha, Qatar in 2012 and 2015, respectively. Her BS was in computer engineering while her MS was in computer networking. She is currently an Assistant Professor of Computer Science, Texas Tech University

From 2012–2015, she worked as a research assistant with Qatar University on an NPRP (National Priorities Research Program) funded project targeting physical layer security. Since 2015, she has been working as a Graduate Research Assistant at Washington University in St. Louis. Her research interests span network security, distributed systems, the Internet of things, and financial technology. She is an author of 1 book chapter, six research articles, and has been a presenter at many international conferences.

Tara Salman is a recipient of the Cisco Certified Network Associate (CCNA) certification in 2012 and has completed all four levels of CCNA at Cisco academy-Qatar university branch.

Ali Ghubaish (Graduate Student Member, IEEE) received the B.S. degree (Hons.) in computer engineering (minor in networking) from Prince Sattam Bin Abdulaziz University, AlKharj, Saudi Arabia, in 2013, and the M.S. degree in computer engineering from Washington University in St. Louis, MO, USA, in 2017, where he is currently pursuing the Ph.D. degree in computer engineering.

From 2013 to 2014, he worked as a Teaching Assistant at Prince Sattam Bin Abdulaziz University. Since 2018, he has been working as a Graduate Research Assistant at Washington University in St. Louis. His research interests include network and system security, the Internet of Things, the Internet of Medical Things, healthcare systems, and unmanned aerial vehicles (UAVs) communications.

Devrim Unal is a Research Assistant Professor of Cyber Security at the KINDI Center for Computing Research, College of Engineering, Qatar University. He obtained his M.Sc. degree in Telematics from Sheffield University, UK and Ph.D. degree in Computer Engineering from Bogazici University, Turkey in 1998 and 2011, respectively. Dr. Unal's research interests include cyber–physical systems and IoT security, wireless security, artificial intelligence and next generation networks. He is a member of IEEE

Abdulla Khalid Al-Ali obtained his Master's degree in Software Design Engineering and Ph.D. degree in Computer Engineering from Northeastern University in Boston, MA, USA in 2008 and 2014, respectively. He is an active researcher in Cognitive Radios for smart cities and vehicular ad-hoc networks (VANETs). He has published several peer-reviewed papers in journals and conferences. He has been awarded the Platinum medal in the Educational Excellence Day Prize for Ph.D. holders in 2015. Dr. Abdulla is currently the Head of the Technology Innovation and Engineering Education (TIEE) at the College of Engineering in Qatar University.

Raj Jain is currently the Barbara J. and Jerome R. Cox, Jr., Professor of Computer Science and Engineering at Washington University in St. Louis. Dr. Jain is a Life Fellow of IEEE, a Fellow of ACM, a Fellow of AAAS, a recipient of 2018 James B. Eads Award from St. Louis Academy of Science, 2017 ACM SIGCOMM Life-Time Achievement Award, 2015 A.A. Michelson Award from Computer Measurement Group and ranks among the most cited authors in Computer Science. Previously, he was one of the Co-founders of Nayna Networks, Inc — a next-generation telecommunications systems company in San Jose, CA. He was a Senior Consulting Engineer at Digital Equipment Corporation in Littleton, Mass, and then a professor of Computer and Information Sciences at Ohio State University in Columbus, Ohio. He is the author of "Art of Computer Systems Performance Analysis," which won the 1991 "Best-Advanced How-to Book, Systems" award from Computer Press Association.