AI-based Intrusion Detection Systems to Secure Internet of Things (IoT)

by

Yazan Otoum

Thesis submitted in partial fulfillment of the requirements for the
Ph.D. in Electrical and Computer Engineering

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

The Internet of Things (IoT) is comprised of numerous devices that are connected through wired or wireless networks, including sensors and actuators. The number of IoT applications has recently increased dramatically, including Smart Homes, Internet of Vehicles (IoV), Internet of Medical Things (IoMT), Smart Cities, and Wearables. IoT Analytics [1] has reported that the number of connected devices is expected to grow 18% to 14.4 billion in 2022 and will be 27 billion by 2025. Security is a critical issue in today's IoT, due to the nature of the architecture, the types of devices, the different methods of communication (mainly wireless), and the volume of data being transmitted over the network. Furthermore, security will become even more important as the number of devices connected to the IoT increases. However, devices can protect themselves and detect threats with the Intrusion Detection System (IDS). IDS typically use one of two approaches: anomaly-based or signature-based. In this thesis, we define the problems and the particular requirements of securing the IoT environments, and we have proposed a Deep Learning (DL) anomaly-based model with optimal features selection to detect the different potential attacks in IoT environments. We then compare the performance results with other works that have been used for similar tasks. We also employ the idea of reinforcement learning to combine the two different IDS approaches (i.e., anomaly-based and signature-based) to enable the model to detect known and unknown IoT attacks, and classify the recognized attacked into five classes: Denial of Service (DDoS), Probe, User-to-Root (U2R), Remote-to-Local (R2L), and Normal traffic. We have also shown the effectiveness of two trending machine-learning techniques, Federated and Transfer learning (FL/TL), over using the traditional centralized Machine and Deep Learning (ML/DL) algorithms. Our proposed models improve the model's performance, increase the learning speed, reduce the amount of data that needs to be trained, and preserve user data privacy when compared with the traditional learning approaches. The proposed models are implemented using the three benchmark datasets generated by the Canadian Institute for Cybersecurity (CIC), NSL-KDD, CICIDS2017, and the CSE-CIC-IDS2018. The performance results were evaluated in different metrics, including Accuracy, Detection Rate (DR), False Alarm Rate (FAR), Sensitivity, Specificity, F-measure, and training and fine-tuning times.

Acknowledgements

First, I praise Allah, the almighty, the most gracious, and the most merciful, for his blessings and the strength he gave me during my studies to complete this thesis.

I also express my sincere gratitude to my supervisor, Prof. Amiya Nayak, whose sincerity and motivation I will never forget. Prof. Amiya has been an inspiration as I hurdled through the path of this Ph.D. degree. He is the true definition of a leader; I learned a lot from him—especially the importance of viewing the "big picture" instead of the minutia.

I am grateful for my father and mother, whose constant love and support keep me motivated and confident. My accomplishments and success are always because of their prayers.

I am deeply grateful to my extraordinary wife, Ala'a; I am deeply grateful for her support and patience during my Ph.D. journey. She gave me unconditional love and support. Now, I'll never stay late on campus. I must also thank my wonderful wife's family for their encouragement.

I thank my sisters (Razan and Rawan), and my brother Yazid for their continual support and prayers that made all the difference.

To my daughter Layan and sons Aoun and Shahm, thank you for being my smile and always making me forget how exhausted I was. My dear daughter was steadfast in her campaign to keep me home from school—finally, you can retreat from this battle.

Finally, I extend gratitude to my family members, friends, and colleagues who support me during this journey.

Dedication

To my parents who always have me in their hearts just as I have them in my heart. Their prayers and support made my journey possible.

To my loving wife, who is my true north and always supports, encourages, and pushes me to be better.

Contents

List of Tables

List of Figures

Acronyms

AE Autoencoders. 17

ANN Artificial Neural Network. 15

CAN Controller Area Network. 80

CNN Convolutional Neural Network. 16, 28, 62, 79, 83, 118

DBN Deep Belief Network. 17, 31, 32, 71, 74, 92, 93, 97, 102, 115, 118, 119

DDoS Denial of Service. ii, 61, 90

DHE Discrete Hessian Eigenmap. 60

DL Deep Learning. ii, 4, 8, 15

DNN Deep Neural Network. 21, 28, 32, 79, 80, 83, 97, 99, 102, 106, 108, 110, 117, 119, 120

DoS Distributed Denial of Service. 90

DR Detection Rate. ii, 61, 72, 99

DRL Deep Reinforcement Learning. 17

DRNN Deep Recurrent Neural Network. 71, 74

DT Decision Tree. 32, 115, 116

ECU Electronic Control Unit. 80

FAR False Alarm Rate. ii, 61, 72

FL Federated Learning. 4, 18, 104

FTL Federated Transfer Learning. 20, 108

GA Genetic Algorithm. 31

GBT Gradient Boosting Tree. 32

GST Generalized Suffix Tree. 61, 66

HMS Human Mental Search. 60

IDS Intrusion Detection System. ii, 2, 4, 14, 21, 36, 37, 60–62, 91–93, 102, 106, 117, 118, 120

IoMT Internet of Medical Things. ii, 34, 104, 106, 117, 120

IoT Internet of Things. ii, 1–4, 8, 26, 28, 34, 39, 40, 56–61, 63, 65, 68, 71, 77, 78, 105

IoV Internet of Vehicles. ii, 34, 80, 90, 91, 102

ITS Intelligent Transport Systems. 8, 80

LR Logistic Regression. 30

LSTM Long Short Term Memory. 17, 21, 28, 118

ML Machine Learning. 8, 15, 104

PADS Position Aware Distribution Signature. 61

PCA Principle Component Analysis. 31

QoS Quality of Service. 91

R2L Remote-to-Local. ii, 61, 68

RBM Restricted Boltzmann Machine. 17, 118

RF Random Forest. 29, 30, 32, 92, 94, 102, 119

RNN Recurrent Neural Networks. 16, 62, 80, 99, 118

ROC Receiver Operating Characteristic. 99, 100, 102, 120

SDPN Stacked Deep Polynomial Network. 47, 51, 118

SGD Stochastic Gradient Descent. 115, 116

SMO Spider Monkey Optimization. 40, 51, 118

SNR Signal-to-Noise Ratio. 61, 68

SVM Support Vector Machine. 9, 32, 62, 80, 99, 115

TL Transfer Learning. 4, 79, 80, 96, 109

U2R User-to-Root. ii, 61, 68

Chapter 1

Introduction

1.1 Overview

The IoT is made up of many different physical endpoints or sensors that are connected to each other and to the Internet. They can collect data from surrounding environments and share it between themselves. Small devices with low power and limited resources can also send data to IoT gateways, where it is aggregated and connected to endpoint networks. As the industry develops and grows, security in IoT is becoming very challenging. This is largely due to the heterogeneity of IoT architecture, the different types of accessed devices, multiple approaches of communication, and the enormous volume of data that are being transmitted throughout the network. Security issues typically involve authentication, data privacy, availability, confidentiality, integrity, energy efficiency, single-point failures to be verified, and more [2]. Most security threats can be categorized as follows:

- High-level threats: insecure interfaces, insecure software/firmware, and middleware security.

- Intermediate level threats: routing disruptions, replay attacks, insecure neighbour discovery, buffer reservation, sinkholes, authentication, session establishment, and privacy violations.

- Low-level threats: jamming, Sybil, spoofing, insecure initialization, and sleep deprivation attacks.

IDS can be used to detect the inside/outside attacks that breach through existing security measures. There are many IDSs in the literature of IoT security issues. With traditional IDSs, data are collected by sensors and are then sent to an analysis engine, which examines the data and detects intrusions. Once an intrusion has been detected, the reporting system generates an alert for the network administrator. IDS approaches can be further classified into signature-based and anomalous-based methods, according to whether the system or network behaviours match an attack signature or deviation from the normal that exceeds the threshold. A signature-based IDS stores a set of attack signatures and identifies arrival attacks by validating their signature in the database. Thus, a signature-based IDS is an efficient tool to identify known attacks in the network because it only uses the signatures that were previously stored. An anomaly-based IDS can predict unknown attacks by monitoring and analyzing the traffic according to the packet features, which can differentiate normal traffic from attack traffic. Signature-based approaches have an advantage over anomaly-based methods because they are simple and can operate online in real-time.

1.2 Motivation

IDS is a popular system that is used to detect network vulnerabilities. The majority of IDSs apply machine-learning algorithms that are efficient in detecting attacks [3, 4].

As discussed earlier, in general, IDSs are either signature-based or anomaly-based. The limitations of each method can be overwhelming when combined and implemented as a single system. The key motivation of this thesis is to develop an IDS system that can predict any potential type of attack for the vast volumes of traffic that enter the IoT networks. We will also explore its application using different machine-learning techniques. Furthermore, we aim to protect these diverse devices that have limited resources by considering the performance metrics (i.e., Detection Rate (DR) and False Alarm Rate (FAR)), ability to support real-time systems, and the required training time.

1.3 Challenges & Problem Statement

Designing and developing an IDS for IoT networks can be challenging, for example:

- Choosing the proper placement for the security device or software in the IoT infrastructure can be difficult (e.g., host-based or network-based). The centralized approach offers the advantage of centralized management but it can also lead to system processing overload, which may affect the Quality of Service (QoS) of the connected devices. In contrast, the distributed strategy can reduce the amount of monitored traffic and increase the processing capacity. However, implementing an IDS in different regions of the IoT network can be difficult because of the associated management issues.

- The limited resources for the connected devices can cause problems, such as memory, data transmission rate (bandwidth), and computational resources.

- The real-time limitation must be considered when proposing an IDS for any security solution.

- The imbalanced datasets problem [5] can be experienced by most of the cybersecurity datasets, in which the model will skew to the normal traffic class and this will make it harder to detect potential attacks.

Using traditional ML/DL methods can have some limitations when it comes to the IDS, for example:

- Achieving higher performance needs a larger amount of data for training, which is considered to be computationally expensive.
- The dataset in traditional machine-learning algorithms needs to go back and forth between the cloud server and the edge devices, which limits real-time learning.
- The final limitation is that the model needs more time for training and to achieve optimal performance.

Considering the previously mentioned requirements of securing IoT environments, in this thesis we propose a DL anomaly-based model with optimal features selection to detect different potential attacks in IoT networks. We combined the two IDS approaches (i.e., anomaly-based and signature-based) to ensure that the model is able to detect both known and unknown IoT attacks. We have also used new learning techniques, such as Federated Learning (FL) and Transfer Learning (TL), to solve the issues related to the need for a large amount of data to train the model, user data privacy, model performance, and learning speed.

1.4 Contributions

The main contributions of this thesis are as follows:

- We have proposed a novel, deep-learning based intrusion detection system that is able to identify severe anomalies. A spider monkey optimization algorithm has been used to extract the most relevant features from the dataset. A stacked deep polynomial network is then applied to identify the optimal features and to classify the data as normal or anomalous in different attack categories (see Chapter 3).

- We have proposed the AS-IDS model, which combines signature-based and anomaly-based IDSs. This model can detect both known and unknown attacks, and reduce high false alarm and false positives rates. The proposed AS-IDS model has a better performance than the Deep-RNN and DBN algorithms when used with the IDSs (see Chapter 4).

- We have used the transfer learning technique to transfer knowledge between two different benchmark datasets (i.e., CICIDS2017 and CSE-CIC-IDS2018). By applying the Deep Neural Network (DNN) and Convolutional Neural Network (CNN) deep learning algorithms, our proposed approach is able to achieve satisfactory performance and reduce the training/fine-tuning time for the target domains (see Chapter 5).

- We have designed a novel federated transfer model that can build a scalable, customizable model out of local models to preserve the local models' privacy and lower training time and memory usage. The proposed model uses a DNN algorithm to train the network and transfer the knowledge from the connected local edge models to build a global customized model without exposing data privacy (see Chapter 6).

1.5 Outline

This thesis is organized as follows. In Chapter 2, we will review the related topics, including IoT architecture and security approaches, different ML/DL techniques used to secure IoT, and state-of-the-art studies for IDS in IoT and its applications. Chapter 3 proposes a Deep Learning-based Intrusion Detection System (DL-IDS) with optimal features to detect IoT environment security threats. In Chapter 4, we propose a model that combines signature and anomaly-based IDS approaches to detect known and unknown attacks in IoT networks. Chapter 5 uses the concept of transfer learning to propose an infrastructure-independent IDS to secure intra-vehicle and external networks. Chapter 6 will propose a Federated Transfer Learning-based IDS to secure a patient's healthcare-connected devices. Finally, Chapter 7 will conclude this thesis and make several recommendations for future work.

1.6 List of Publications

Journals:

- Otoum, Y., Chamola, V., Nayak, A. Federated and Transfer Learning-Empowered Intrusion Detection for IoT Applications, IEEE Internet of Things Magazine, 2022. (Accepted)
- Otoum, Y., Nayak, A. AS-IDS: Anomaly and Signature Based IDS for the Internet of Things. J Netw Syst Manage 29, 23 (2021). https://doi.org/10.1007/s10922-021-09589-6
- Y. Otoum, D. Liu, A. Nayak, "DL-IDS: a deep learning–based intrusion detection framework for securing IoT", Transactions on Emerging Telecommunications

Technologies, pp. 1-14, 2019.

Conferences:

- Y. Otoum, S. Yadlapalli, and A. Nayak, FTLIoT: A Federated Transfer Learning Framework for Securing IoT," IEEE GLOBECOM 2022. (Accepted).
- Y. Otoum, Y. Wan and A. Nayak, "Transfer Learning-Driven Intrusion Detection for Internet of Vehicles (IoV)," 2022 International Wireless Communications and Mobile Computing (IWCMC), 2022, pp. 342-347.
- Y. Otoum and A. Nayak, "Signature-Over-The-Air with Transfer Learning IDS for Intelligent Connected Vehicles (ICV)," 2021 IEEE Globecom Workshop, 2021.
- Y. Otoum, Y. Wan and A. Nayak, "Federated Transfer Learning-Based IDS for the Internet of Medical Things (IoMT)," 2021 IEEE Globecom Workshop, 2021.
- Y. Otoum, A. Nayak, "On securing IoT from Deep Learning perspective", 2020 IEEE Symposium on Computers and Communications (ISCC), pp. 1-7, 2020.

Chapter 2

State of the Art

2.1 Introduction

The extensive growth of the IoT has impacted diverse applications, including smart homes and cities, Intelligent Transport Systems (ITS) and smart factories [6]. IoT integrates billions of smart devices -predicted to increase from 27 billion in 2017 to 125 billion by 2030- and manages communication between them. This degree of expanded connectivity requires extensive further analysis with respect to security, and the involvement of millions of factors and users increases vulnerability in IoT environments. However, DL approaches, which originated from Machine Learning (ML), have been efficient in many research fields, and current studies show the effectiveness of ML/DL for IoT security applications. In this chapter, we present detailed analyses of IoT security requirements and challenges, discuss the specific role of ML/DL and review state-of-art research work in IoT environments using ML/DL approaches. We also performed a comparative analysis of ML/DL algorithms such as RNN, LSTM, CNN, DBN and AE and techniques such as federated and transfer learning. And finally, we identified research issues in the current investigations and outlined the future directions of ML/DL algorithms in IoT

security domains. Machine learning algorithms, such as K-nearest neighbour (KNN), Naïve Bayes, Support Vector Machine (SVM), linear regression, regression trees, random forest, K-means and deep learning algorithms have proven to be effective in IoT environments [7]. ML and DL algorithms are typically used in IoT for Big Data analytics such as classification, cluster formation and diagnosis, and end-to-end security for authentication, key agreement and anomaly detection.

2.2 Background

This section provides a brief overview of IoT Architecture, IoT security threats, and discusses potential Machine and deep learning approaches.

2.2.1 Internet of Things (IoT) Architecture

The IoT architecture [8] consists of three layers, known as application, network and perception layer [9], as shown in Figure 2.1. The application layer is the top level in the architecture, and it provides services to users such as smart cities and smart grids. The network layer is the most critical, as it integrates the connected devices using the infrastructure, including gateways and switches, with different communication technologies and protocols, such as Wi-Fi, ZigBee and Bluetooth. The perception layer, also known as the sensor layer, collects and processes the data from sensors, and can also control physical objects in the near environment, such as actuators. The perception layer has two sections: sensor nodes like RFID tags, and wireless sensor networks that are self-organizing and have many sensor nodes deployed in large areas.

The IoT Gateway is also very important, as it is responsible for data forwarding, node control and communication protocol compatibility. It has a wide range of access capa-

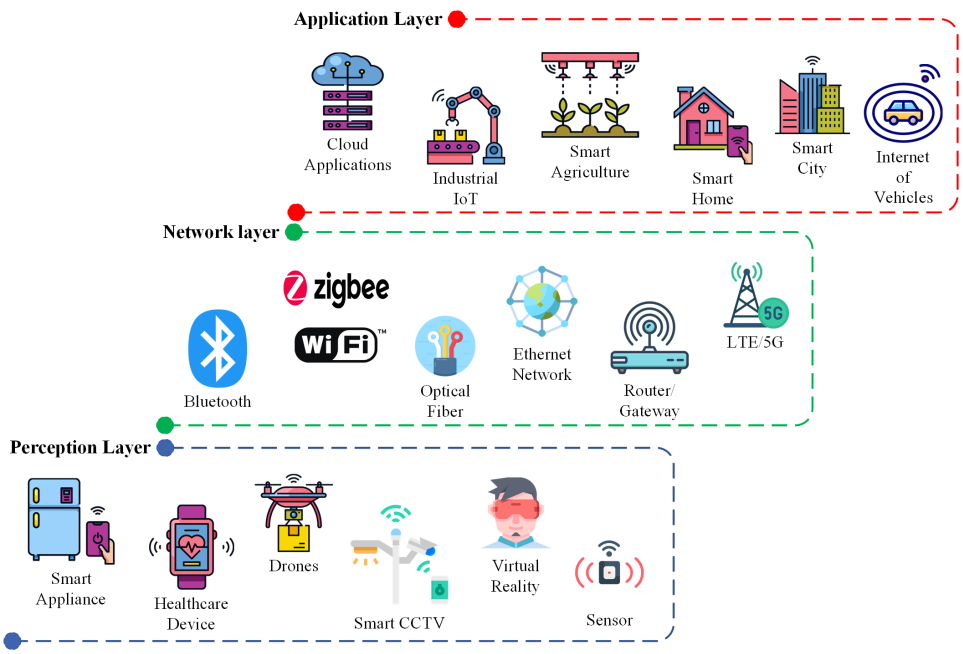

Figure 2.1: IoT Layered Architecture

bilities, including seamless manageability and protocol interworking between traditional networks and WSN. Manageability Protocol interworking supports the traditional network and WSN seamlessly. IoT enables interconnection between many types of devices, including smart sensors, terminals and mechanisms, industrial systems, autonomous vehicles and machine systems. These range from simple wearable accessories to large machinery [10–12]. Many criteria are involved to establish communication among these "Things", including WiFi\IEEE 802.11, WiMax\IEEE 802.16, LR-WPAN\IEEE 82.15.4 and ZigBee\IEEE 802.15.4, as well as cellular standards such as Long Term Evolution (LTE) and the fifth generation (5G) networks. The enormous volume of data generated by many interconnected devices, often called "Big Data", is transmitted to remote cloud storage systems and is accessible by end users through cloud services.

Due to rapid advances in IoT infrastructure development, monitoring security has become increasingly complex and challenging. Malicious software, viruses and hackers

are significant threats as they can degrade data integrity. Insecure data also reduces IoT security, thereby inciting additional risk. In addition, wireless communications can encourages eavesdropping by unqualified entities. Hence, security improvement and transformation are critical to the IoT field. A typical IoT architecture consists of four layers: a device layer, connectivity layer, support layer and application layer [13]. However, the interactions of vast numbers of things and users raises many IoT challenges in each layer, including architecture interoperability and accountability, privacy and security issues.

2.2.2 IoT Security

Each layer in an IoT system is susceptible to threats and should be secured by one or more techniques of effective anomaly detection, such as authentication and encryption [14]. Table 2.1 indicates the IoT security threats at each layer. Here, we discuss network layer threats and attacks on IoT systems. DoS attacks are the most common, as they render network services unavailable to authorized users by overwhelming all resources with a massive amount of traffic [15]. The man-in-the-middle attack is a malicious device controlled by the attacker that can intercept communications between two authorized devices by presenting itself as a store. Then it can forward data with the intent to modify, eavesdrop and control the communication between the users.

, Malware challenges, Phishing attack, Malicious virus/worm, Malicious scripts.

In a spoofing attack, the attackers can control the address of a valid IoT device, and thereby interfere with communications between other network devices; they can also send malicious data that appears valid. Security issues are considered particularly important, since security provisioning in IoT environments is critical due to heterogeneity, scalability, vulnerable technologies, resource limitations, mobility, diversity of attack sources, lack of standardization and data sensitivity [2, 16]. An effective security mechanism must

Table 2.1: Attack Classifications

Layer	***Security threats***	***Threats description***
Perception Layer (Device)	Node capture attacks, Malicious code injection attacks, false data injection attacks, Replay attacks, Cryptanalysis attacks and side channel attacks, eavesdropping and interference, and Sleep deprivation attacks.	The security threats in the perception layer focus on forging aggregated data from the IoT devices, and destroying perception devices[9].
Network Layer (Connectivity)	DoS attacks, Spoofing attacks, Sinkhole attacks, Wormhole attacks, Man in the middle attack, Routing information attacks, Sybil attacks, and Unauthorized access.	The security threats in this layer focus on affecting the availability of the network resources, since the main purpose of the layer transmits aggregated data. However, most devices in the IoT environment are connected wirelessly [14].
Application Layer	Phishing attack, Malicious virus/worm, Malicious scripts, Data privacy, and Access control.	The security threats in the application layer are focused on software attacks, since the main purpose of the layer is to support services requested by the users[9].

satisfy the following criteria:

1. Confidentiality: Ensures that the data is not accessible to unauthorized users, entities and processes.

2. Integrity: Ensures that the data has not been modified by an untrusted third party.

3. Authentication: Ensures that data sources and data users are authorized entitles through effective verification.

4. Non-repudiation: Ensures that the data owner cannot deny having sent the data.

5. Availability: The data must be available to legitimate users at any given time.

6. Privacy: Both data and users should not be traceable by malicious users based on their behaviour.

As mentioned previously, each layer in an IoT network is vulnerable to multiple security threats. Attacks increased by 217.5% in 2018, from the 10.3 million logged by SonicWall in 2017 to 22.4 million in their 2019 Cyber Threat Report. It is critical to secure each layer in an IoT through effective authentication, encryption, anomaly detection and so on [14, 17]. Some significant attacks can be described as follows:

- Eavesdropping: This is often known as a snooping attack, as an intruder tries to intercept private communication between two legitimate users.
- Jamming: This is a type of denial of service, as it overwhelms wireless link resources by sending many unwanted packets to prevent other nodes from using the channel.
- Sybil attack: This attack involves an intruder subverting the reputation system of a peer-to-peer network by creating numerous fake identities.
- Replication attack: This is a network-oriented attack in which an attacker replicates a node to overhear the legitimate communication.
- Flooding attack: This is a type of DoS attack launched by intruders to bring down a network or service by sending an excessive overload of traffic.
- DoS attack: This is a severe cyber-attack intended to make a network resource or service unavailable to legitimate users. It can be launched in a distributed manner with multiple attacking nodes [18].

In IoT, the device layer is comprised of a huge number of sensor nodes, and the connectivity layer supports communication among these nodes. The interconnected architecture of IoT and major security issues is depicted in Figure 2.2.

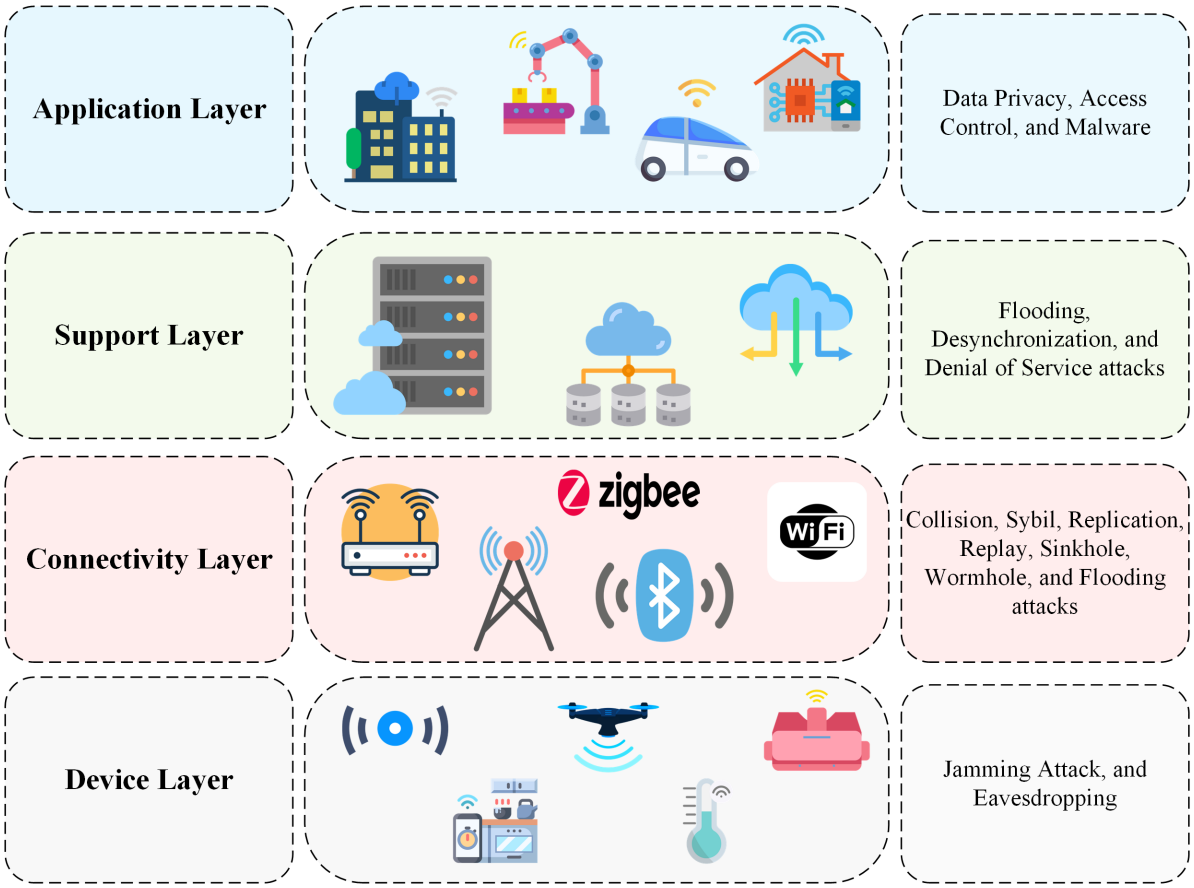

Figure 2.2: IoT Environment With Security Issues

2.2.3 Machine and Deep Learning for IoT Security

The significant problems with IoT security are related to authentication, access control, user privacy, data privacy and intrusion detection systems, and packet feature extraction is also critical for anomaly detection. In ML/DL-based anomaly detection, IDS methods, data pre-processing, feature extraction and optimal feature selection are the significant factors that determine IDS accuracy. Other common problems with DL-based approaches are the complexity, and the time required for training. Canziani et al. analyzed several DNN models, and found that minor increases in accuracy require extensive time [19], and that parameter tuning is also a major problem since the number of layers and

accuracy have a clear relationship. Thus, many hyper parameters are required for optimal initiation of DL methods, which are sensitive to the structure and the size of data. Research problems in ML/DL-based IoT security environments include:

- Providing end-to-end security (authentication, access control, privacy, integrity and IDSs);
- Data pre-processing, optimal feature extraction and selection;
- Parameter tuning Optimization; and,
- Decreasing time and complexity.

DL is a state-of-the-art ML field with powerful analytics capabilities, used for applications such as computer vision, bioinformatics and natural language processing. It is seen to demonstrate significant performance improvement over some ML algorithms and has recently been used in some IoT applications in edge/fog computing. These applications require substantial volumes of data to be transferred over the network, and DL gives better performance with extensive data than ML. Besides, DL can work with new features that can be used to solve the problems without human interactions [20].

DL approaches for IoT are categorized as supervised DL methods, unsupervised DL methods and hybrid DL methods [21]. DL methods are typically based on many layers of an Artificial Neural Network (ANN) that can learn hierarchical representations in deep architectures. The significant feature of DL methods over traditional ANN is that they can discover hidden features from raw data, and each layer is trained on a set of features learned from previous layer outputs. This means that the inner-most layers can learn complex features through aggregation and recombination. The general architecture for training of DL model is illustrated in Figure 2.3.

Some of the primary ML and DL methods are explained as follows [22–24]:

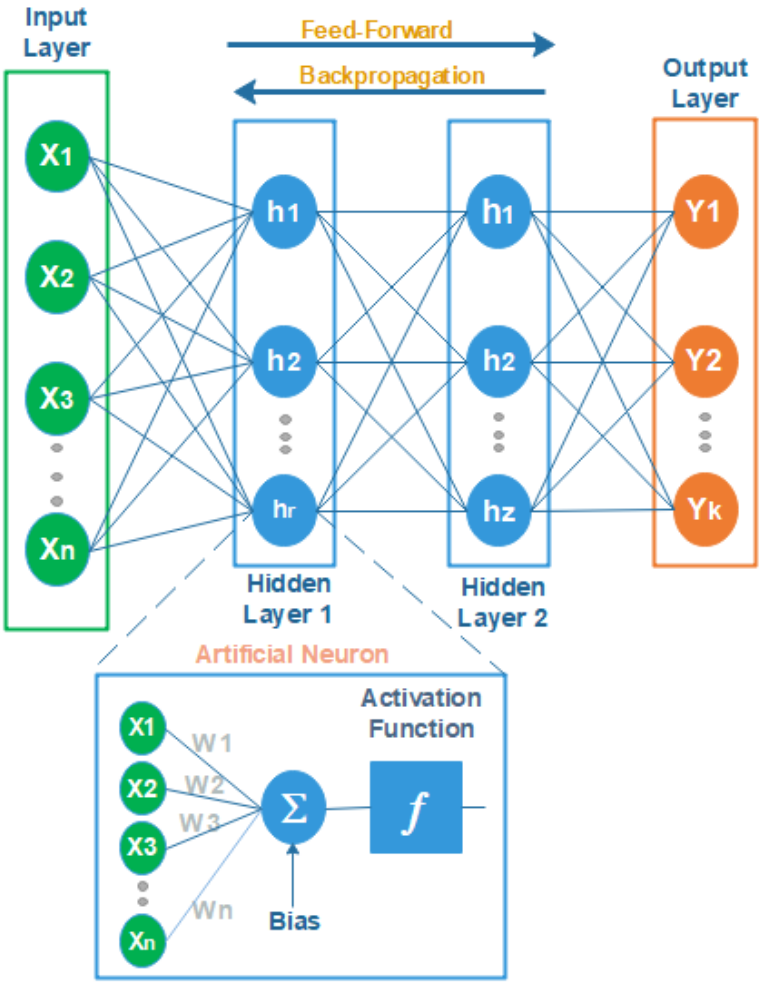

Figure 2.3: Training of the DL model

1. Convolutional Neural Network (CNN): CNNs are designed to support translation equivariance computations by extracting high level features from a series of hidden layers. The hidden layers are made up of convolutional layers that consist of a set of learnable parameters. The spatial size of representation is reduced by pooling layers to minimize both the number of parameters and computation time.

2. Recurrent Neural Networks (RNN): RNNs are introduced to analyze previous samples and classify them, which is significant for prediction applications. With RNN,

each neuron has a feedback loop that returns current output as input for the next step.

3. Long Short Term Memory (LSTM): LSTM is a an extension of RNNs in which each neuron is equipped with multiplicative forget, read and write gates to store feedback information. The neuron stores and removes feedback information based on the gate operations.

4. Autoencoders (AE): AEs are built with equal numbers of input and output units with one or more hidden layers. With AEs, the input is reconstructed through simple transformations to solve unsupervised learning problems and transfer learning. Recently, several variants of AEs have been developed, such as denoising, contractive, stacked, sparse and variational AEs.

5. Restricted Boltzmann Machine (RBM): RBM is a stochastic ANN comprised of a visible layer and a hidden layer. The term 'restricted' is related to neuron connectivity. RBM builds a bipartite graph in which each neuron in the visible layer has connectivity with each neuron in the hidden layer, and vice versa. However, RBM is restricted only to connections between neurons in the same layer.

6. Deep Belief Network (DBN): DBNs are a type of generative ANN with a visible layer and several hidden layers. Each layer is treated as an individual RBM and trained on top of the previous layer. The hidden layers can extract hierarchical representation of the training data, and reconstruct their input data.

7. Deep Reinforcement Learning (DRL): DRL combines DNN and reinforcement learning approaches to manage many states and long-term rewards in the RL approach. With DRL, RL determines the best actions over a set of states in the environ-

ment designed by the DNN model. In Table 2.2, a brief analysis on ML and DL approaches is provided along with their IoT applications.

2.2.4 Transfer Learning (TL)

Transfer learning is when previously learned knowledge such as features and weights is reused for a new related problem. Transfer of trained models from one domain to another involves moving models that were trained on one dataset to another dataset. If we have a domain D defined by the features space x and marginal probability $P(X)$, where X is a sample data point. Thus, $D = \{\chi, P(X)\}$, where a task, T, can be represented as a two-element tuple of the label space, Y and objective function n that can be expressed as $P(\gamma, X)$ in probabilistic terms. For a given domain D, a task is defined by $T = \{\gamma, P(Y|X)\} = \{\gamma, n\}$ and $Y = \{y_1, ..., y_n\}$, $y_i \in \gamma$. Therefore, transfer learning can be expressed for a given source domain D_s, a corresponding source task T_s, a target domain D_T, and a target task T_T. Consequently, by combining the information about the target conditional probability distribution $P(Y_T|X_T)$ in D_T with information about D_S and T_S, where $D_S \neq D_T$ or $T_s \neq T_T$, we can ensure transfer learning. TL has different approaches based on the technique used to transfer the knowledge, such as Inductive Transfer Learning (ITL), Unsupervised Transfer Learning (UTL), and Transductive Transfer Learning (TTL). In ITL, the source and target domains (feature spaces) in the two models should be similar, but the source and target tasks (label spaces) should differ. However, the ITL is of two types: multitask learning and self-taught learning.

2.2.5 Federated Learning (FL)

In FL approach, a global machine learning model can be trained and reused with local data on the edge side without exchanging their data. The new locally trained model

Table 2.2: ML and DL approaches along with their IoT applications

Model	**Category**	**Learning model**	**Characteristics**	**IoT application**
CNN	Discriminative	Supervised	Convolution layers play vital role in computations Less connection compared to DNN Requires large amount of dataset	Plant disease detection Traffic sign detection
RNNs	Discriminative	Supervised	Processes sequence of data using internal memory Works well with time-dependent data	Movement patter prediction Behavior detection
LSTM	Discriminative	Supervised	Access memory cells and protected by gates Performs well with data of long time lag	Human activity recognition Mobility prediction
AEs	Generative	Unsupervised	Has same number of input and output units Works well with unlabeled data and output reconstructs the input	Machinery fault detection Emotion recognition
GAN	Generative and Discriminative	Unsupervised	Generate a new data (e.g adversarial attacks) which can be used to improve the detection systems	Detecting anomalous behavior using GAN based-Intrusion Detection Systems IDS for IoT
RBMs	Generative	Unsupervised, supervised	Supports feature extraction, dimensionality reduction, and classification Has expensive training procedure	Indoor localization Energy consumption prediction
DBN	Generative	Unsupervised, Supervised	Greedy training of the network layer by layer Suitable for hierarchical features discovery	Fault detection classification Security threat identification
DRL	Generative	Reinforcement learning	No supervision is required and only a reward signal is used Based on sequential decision making	Fault detection and isolation in IIoT Supply-chain risk management Factory automation DDoS attack prevention

can be sent back to be aggregated with the global model (i.e. taking the average of the locals model weights), and then a combined and experienced global model is sent back to the edge-connected devices. FL has recently garnered much attention in many IoT applications because it preserves data privacy [25]. It has three primary approaches [26]: Horizontal Federated Learning (HFL), Vertical Federated Learning (VFL), and Federated Transfer Learning (FTL). The instance when the edge devices share a few or all of the features' set is referred to as HFL. However, the instance when the edge device shares a few or all of the features set with the global model is referred to as VFL. The authors in [27] proposed a model that used the Vertical Federated Learning technique to secure a cloudlet-based recommendation system for Electric Vehicles (EVs), such that the EVs data will not be shared outside the platforms. Blockchain technology is integrated with their model for generating a trusted network of cloudlets responsible for transferring locally-trained parameters. According to the simulation results of their proposed recommendation system, EVs are more optimally distributed over an area with charging stations. The authors found that the decentralized recommender system with ten cloudlets is 5.2 sec faster than a typical centralized recommendation system. In FTL, the parameters of the trained global model are reusable with the connected local models without breaching user data privacy. The global model can be combined with recent models trained on the edge-side (Averaging the local model's weights). Afterwards, the aggregated global model could be transferred to the edge devices. Using those types of learning approaches will perceive the sensitive data on the edge side, reduce the network latency, and minimize the run of the cloud resources and hardware use.

2.3 Survey on Machine and Deep Learning-based IDS to secure IoT

The DL and ML methods were developed for signal authentication in IoT environments [28, 40]. Ferdowsi et al. proposed a LSTM structure to extract sets of stochastic features from signals generated by IoT devices. The features were then watermarked to manage cyber-attacks in IoT environments, and the dynamic feature extraction also supported detection of eavesdropping attacks. However, this method is not suitable for very large IoT environments, since authenticating all IoT devices in a centralized cloud server can become overly complex. To overcome this issue, the LSTM-based signal authentication method was combined with game theory approaches to derive a mixed strategy Nash Equilibrium (NE) game. This method could be capable of handling numerous IoT devices, but it also has high complexity which is not favorable for IoT environments.

In [30], Using IDS, Deep Neural Network (DNN) was adapted to classify significant attacks in IoT environments. The performance of DNN was evaluated using cross-validation, repeat cross-validation and subsampling, and the DNN parameters were initialized by a grid search method. It was concluded that DNN performed well for diverse datasets such as imbalanced and biased results, but the learning parameter determination takes longer for grid search.

Diro and Chilamkurti [31] proposed a DL approach to detect the distributed attacks, including probe, R2L, U2R and DoS on Social IoT (SIoT) over fog nodes. The analysis showed that the DL approach outperforms other ML methods, though it requires further analysis concerning network payload-based intrusion detection. The performance was compared with conventional ML and distributed detection techniques and validated against the centralized detection method. The performance of distributed detection was

Table 2.3: Comparison on previous works

Research work	**Purpose**	**Features**	**Pitfalls**
LSTM [28]	Signal authentication	Stochastic features are extracted form signal Signal features are watermarked for authentication	Involvement of centralized server increases complexity Not able to support massive IoT environment
LSTM-NE [29]	Signal authentication	Supports vast number of IoT devices	Introduces high complexity
DNN-IDS [30]	IDS in IoT environment	Works well with different types of datasets Classifies significant attacks	Increases time consumption in parameter tuning since it uses grid search
DL-SIoT [31]	Distributed attack detection in IoT	IDS is distributed among fog nodes Detects probe, R2L, U2R attacks	Intrusion detection is not efficient Further analysis is needed on payload based detection
Dense Random Network [32]	Network attack detection	Extracts attack oriented features UDP flood, TCP SYN, sleep deprivation, and broadcast attack are considered	Parameter tuning in dense random network is significant
Deep-Q-Networks [33]	Securing healthcare data	Maintains authentication, access control, and intermediate attacks	Attack detection using non-optimal features affects the accuracy
Deep Eigenspace Learning [34]	Malware detection in IoBT	Uses OpCode sequence for malware detection Feature extraction by graph based method	Increases time consumption Introduces high computational complexity
ML method [15]	DDoS detection in IoT environment	Stateless packet features and stateful packet features are extracted	Not able to handle high-dimensional features Classification accuracy is low
N-BIoT [35]	Network based botnet detection	Features are collected in five time windows	Static features only considered which degrades the performance Accuracy of AE also low due to ineffectual feature extraction
BLSTM-RNN [36]	Botnet detection in IoT	Word embedding is used for text recognition	Suitable only for botnet with limited number of attack vectors
AE-IDS [37]	IDS in fog enabled IoT	Fog based IoT improves scalability and latency	Runtime of AE is large since it is performed based on all features
Deep AE and Deep FNN [38]	IDS in IIoT	Processes feature transformation and feature normalization	Not able to handle other protocols in IIoT
DL based ML [39]	Routing attacks in IoT	Detects routing attacks such as hello flood, decreased rank, version number modification	Not able to detect other severe attacks Extraction of limited number of features limits the attack detection performance

tested with different machines used for network training, and the effectiveness of DL in detecting intrusions was evaluated against conventional IoT learning techniques. Results demonstrated that the distributed framework outperformed the centralized framework and achieved a detection accuracy of 96% to 99%. Further results showed that DL methods had higher accuracy than the conventional learning methods and a lower false alarm rate of approximately 0.85% compared to ML methods. Using DL methods, a recall and an average recall of 99.27% and 96.50% were achieved, respectively. In contrast, ML had a recall and average recall of 97.50% and 93.66%, respectively. These results verified that DL techniques have a high potential to detect cyber-attacks in distributed IoT environments. The experimental analysis found that distributed detection schemes could quickly identify the cyber-attacks compared to the centralized methods due to sharing variables that can prevent the local minima during training. The work could be extended by comparing the distributed DL methods with distinct conventional ML methods using different datasets. Techniques for investigating the payload data of the network for identifying intrusions using critical patterns could also be studied further.

Dense random networks were used to design a deep learning approach to detect network attacks [32]. In this approach, UDP flooding, TCP SYN, sleep deprivation, barrage and broadcast attacks were considered, and were detected based on following metrics:

- UDP flood: The number of outgoing ICMP "destination unreachable" packets.
- TCP SYN: The number of half-opened TCP connections.
- Sleep deprivation: The number of data packets over a long period.
- Barrage attack: The number of data packets over a short period.
- Broadcast attack: The number of broadcast messages.

Based on these metrics, attacks against IoT gateways were detected using the dense random network-based DL approach. The significant issue with this method is parameter tuning of the dense random network, since non-optimal parameters can lead to inaccurate classification. In IoT healthcare environments, security and privacy was maintained using layering based deep-Q-networks [33], which ensured authentication and access control and managed intermediate attacks in the IoT environment. Initially, packet features such as IP address, protocol, file type, frame length, frame number and host post number were extracted from the incoming packets and stored in a local database. The medical data was then classified by deep-Q-networks based on the extracted features, and the classification was performed based on the packet features since extracting optimal features from data packets provides higher accuracy. A deep Eigenspace learning approach for malware detection was presented in the Internet of Battlefield Things (IoBT) [34], where the Operational Code (OpCode) device sequence was applied for malware detection. The device OpCodes were transmuted into vector sequences, and the deep Eigenspace learning approach was applied to identify malicious and benign applications. The entire process started with constructing a graph for each sample, then classifying the sample. This method involves substantial computation, which limits performance for large datasets.

DDoS attack detection in IoT environments was conducted using specific characteristics of IoT-specific network behaviours, such as limited endpoints and regular time intervals between packets [15]. The DDoS detection was performed with KNN, SVM with linear kernel, decision tree and neural network. Two types of stateless features were applied for classification: those with inter-packet intervals, packet sizes and protocols, and those with bandwidth and a number of destinations (IP) were extracted from the data packets. The authors concluded that ML algorithms work well with low-dimensional features, but are not suitable for features that are high-dimensional; deep learning ap-

proaches for anomaly detection are more suitable for managing with high-dimensional features.

A Network-based detection of IoT botnet (N-BaIoT) method based on deep autoencoders was also proposed [35]. With N-BaIoT, features such as packet size, packet count and packet jitter were collected for five different time periods. However, this method only considers static features which limits the performance of autoencoders, and it requires optimal feature selection to improve autoencoder accuracy. Bidirectional LSTM-RNN (BLSTM-RNN) was used for IoT botnet detection in [36], and word embedding was applied for text recognition and attack packet conversion to integer format. Though BLSTM-RNN is suitable for detecting botnets with limited attack vectors, it becomes inefficient when the number of attack vectors increases. Autoencoder, a stacked unsupervised DL approach, was introduced for intrusion detection in fog enabled IoT environment [37], and fog based IoT architecture was designed to improve the scalability of attack detection and minimize latency. The authors concluded that deep learning algorithms outperform shallow learning algorithms for attack detection; though accuracy does improve stacked autoencoder also increases the system runtime. In Industrial IoT (IIoT), network intrusion detection systems are used to protect a network from multiple security threats [38]. Deep autoencoder and deep feed forward neural network are used for intrusion detection, and the overall process involves feature transformation and normalization. This method requires further analysis to learn how to manage different protocols for intrusion detection in IIoT. Yavuz et al. [39] proposed a deep learning-based machine learning algorithm to detect routing attacks in IoT environments. Significant routing attacks, including decreased rank, hello flood and version number modification, were detected and studied based on the following features: average reception time, number of packets received, transmission rate, average transmission time, number of packets

transmitted and total transmission time. Exclusion of features limits detection performance and the ability to detect other severe IoT attacks. Analysis was conducted using the Cooja simulator for IoT networks with 10 to 1,000 nodes, and the method efficiently detected routing attacks in an IoT environment. Further analysis was conducted using the 64.2 million attack datasets generated by actual sensor code and protocol implementation, and attack datasets with routing IoT values were used in the study. Though Deep Neural Network frameworks modelled using the trained datasets had higher precision, recall and accuracy rates, the work was limited due to the unavailability of relevant datasets and poor quality of available information. The work could be extended to study scenarios with diverse rates of normal nodes and suspicious nodes and by incorporating additional features in a single framework to identify multiple types of attacks.

Thamilarasu and Chawla [41] proposed a technique for detecting intrusions in IoT environments using deep learning techniques. Their proposed approach successfully detected suspicious attacks in IoT networks, and the method also provided services such as security and interoperability between distinct communication protocols. The feasibility of ML methods for intrusion detection was examined, and anomalous activities on insecure networks were effectively identified. Optimal solutions for validating performance against intrusion detection were applied to six distinct intrusion scenarios, namely sinkhole, Blackhole, opportunistic service, wormhole, denial of service and Blackhole attacks, and the results indicated a 95% precision rate and 97% recall rate for the intrusion scenarios. In addition, the results demonstrated that higher F1-scores for these five intrusion scenarios were achieved, indicating the efficacy of the detection performance of the proposed method. Analysis of experimental results found that the proposed scheme can be applied practically, and it can potentially identify attacks in IoT environments. The results also demonstrated that real-time intrusions were effectively recognized by

the proposed technique. This research can also be extended to investigate other methods of detecting different IoT intrusions. Techniques related to device cloning and spoofing could be explored for their ability to identify attacks that are location dependent, and methods to track components and estimate journal entries related to detecting direct, isolated and neighbour threats could also be assessed.

Arrington et al. [42] proposed a technique for detecting anomalies using IoT that depended on the behaviour of anomalies within a smart vicinity, while immunity inspired techniques were applied to differentiate extracted behavioural patterns from accurately matching and deviant behaviours. Accurate behavioral frameworks were developed by monitoring every preferred activity through simulations, and behavioural identities were represented numerically by capturing the activity of IoT sensors, combined with event sequences and simulated states. Construction of behavioural models through system monitoring was cost effective and easily evaluated, and the proposed method determined malicious activities within a smart environment using IDSs to recognize behavioral patterns. Suspicious activities were captured by IDSs through IoT sensors within the smart environment. Enhancing the detection sensitivity for identifying peculiarities between behavioural patterns via the IoT components was the prime motive of this research. Techniques for detecting specific activities and the generalization of effective differentiating behaviour patterns should also be investigated. Alghuried [43] proposed a technique to identify application layer intrusions against IoT devices using a unified decision tree and Inverse Weight Clustering (IWC). The data is initially acquired from the IoT components, then transmitted via the gateways to the central unit for processing. Anomalies in data transmitted by IoT components are detected in central units at the application layer. The analysis was conducted using training and testing datasets. The training set consisted of labelled records and resulted in 66% of the dataset, and the testing set was

made up of unlabeled records signifying a dataset of 34%. These results indicate that the proposed method achieved an accuracy of 97% for detecting normal, anomalous and false classifications at a rate of of 2.1%. However, this method can only be applied to numerical and real data not textual data, and it's capabilities are limited to anomaly detection only at the application layer. Thus, attacks altering the software or hardware configurations of IoT components cannot be detected with this method.

DL has been used for many IoT applications for the reasons discussed above. Lane *et al.* [21] used two of the most common deep learning algorithms (i.e. CNN and DNN) to build a model that processes sensor data using three different types of IoT hardware employed in wearables and smartphones. DL approach-based authentication using LSTM that learns the hardware imperfections of low-power ratios has also been proposed for IoT environments [29]. LSTM was designed to be sensitive to signal imperfections in device identification. However, the method is only helpful for device identification as it cannot detect other significant attacks.

LSTM has also been applied for botnet detection in IoT [44]. In addition, an LSTM-based IoT threat hunting approach involving OpCode extraction, feature extraction and deep threat classification was proposed. Though optimal feature selection based on term frequency and inverse document frequency was also tested (TF-IDF), it is only usable with small datasets and is unsuitable for real-time applications. The learning-based deep-Q-network was proposed to maintain security and privacy in IoT healthcare environments [33] by managing authentication, access control and intermediate attacks in IoT. However, attack detection based on non-optimal features leads to degradation of classification accuracy. Botnet detection on the Internet of Battlefield Things (IoBT) was performed using the deep Eigenspace learning approach [34]. The operation code (OpCode) sequence of devices was used for malware detection before classification, and

a feature-based graph was constructed for each sample. As the involvement of large computations results in high computational complexity, this is ineffective for IDS. For botnet detection, a bidirectional LSTM-RNN method was used in IoT [36], and word embedding was applied for text recognition and to convert the attack packet into integer format. However, this method only performs well with a limited number of attack vectors; when the number increases, it becomes ineffective. Stacked AutoEncoder (AE) is an unsupervised DL method proposed to detect intrusion in fog enabled IoT environments [37], and its attack detection latency and scalability were improved by using fog computing in IoT. However, the running time of AE is relatively long, and the detection accuracy is low due to non-optimal features.

In [45], an IDS was designed with the deployment of honeypots, and it was responsible for monitoring the network devices. It computed the belief, disbelief and uncertainty of each node's reputation from those that were managed by the honeypots. As a result, this IDS system was required to incorporate multiple honeypots in the IoT, which could be compromised and make the instruction detection more complex.

Shared model-based hybrid intrusion detection was performed on the IoT botnet dataset [46]. This involved four different phases to detect intrusions: preprocessing, feature selection and signature and anomaly-based IDSs. Features were selected using an information gain-based algorithm, then transmitted to the signature-based IDS model where a C5 classifier was applied. With the anomaly-based model, a one class SVM was used to detect intrusions. In this work, a one class SVM is unable to handle immense dimensional datasets, as it tends to decrease the detection rate. Then, for the newly detected attacks it generates an attack signature for further processes.

In [47], a combination signature and anomaly-based IDS was proposed that used Random Forest (RF) and Boruta algorithms. In the work, the RF computes the Z-

score value for each feature using entropy and the Gini index, and then the attacks are classified according to the default RF parameters. The use of RF was based on the tree, which occupies large memory and cannot process classification results quickly. In [48], the authors proposed a hybrid multi-model solution that utilizes an Ensemble model with stacked generalization. In this work, RF, Logistic Regression (LR) and k-NN are used for training purposes, and SVM based-Stacking was applied for testing. As a result, the SVM classifier algorithm is unable to process effectively when the dataset is in IoT systems, as they have a vast volume of traffic which renders SVM unsuitable. In [49], k-NN and K-means clustering algorithms were proposed for a hybrid IDS. Initially, the pre-processing process for the dataset was performed and then K-means and k-NN were used for clustering and classification, respectively. Normal and attack data types were clustered based on the computed centroid and the distance between points, and from the clustered data k-NN was applied for classification. However, in K-means the selection of the K-value is critical, since if it fails then the clustering process will be inefficient.

A misused detection-based model scheme (Signature-based) was used in [50]. In their work, KDD'99 and UNSW-NB15 datasets were used to detect intrusion, based on the attack signatures. The authors used the kernel principal component analysis algorithm to reduce features' dimensionality and extract significant features, while the Extreme Learning Machine (ELM) algorithm was applied for intrusion detection. This had hybrid kernel functions such as Radial Basis Function (RBF) and polynomial kernel for detection, and the parameters of the ELM were optimized using the Differential Evolution (DE) algorithm with the Gravitational Search Algorithm (GSA). The main drawback of this research was that the Kernel Principal Component Analysis (KPCA) was used to reduce the features' dimensionality by selecting optimal features before the intrusion detection process. Thus, the results indicate low accuracy during the intrusion

detection, since the size of the kernel matrix increases quadratically as the dataset size increases.

The optimization algorithms were also used in the intrusion detection systems, and as in [51], intrusion attacks are detected using the features selection-based algorithms. The Pigeon Inspired Optimization (PIO) algorithm and continuous binary cosine models were applied to select the significant features, and to detect attacks from the dataset. Optimization with AdaBoost machine learning algorithms to detect network intrusions were proposed in [52]. The model first preprocesses the data packet, then performs feature selection using the Artificial Bee Colony (ABC) optimization algorithm. This requires frequent estimates of the fitness values, which could lead to increased processing time. In [53], three different processes were proposed: preprocessing, feature selection and classification. Preprocessing was done by converting the data in the dataset into respective numeric values, and Principle Component Analysis (PCA) was used to select the optimal features and the Genetic Algorithm (GA) based DBN classifier was applied. The GA was used to select the optimal parameters for the DBN algorithm, though it takes more time for optimal selection. In [54], the authors proposed a multi-objective optimization process to minimize the rates of false positives and negatives through detecting a group of generated alerts from various IDSs. The model has four phases; In the first phase, the low-level alerts are classified into meta-alerts for each IDS. Then, the featured meta-alerts are filtered into a set of P-FNs. Next, a clustering step inter-IDS is performed to groups similar meta-alerts together to avoid redundancy. In the last phase, a Binary Multi-objective Optimization Problem (BMOP) is used to detect FNs and FPs. The proposed model is evaluated using real network traffic, NSL-KDD, and DARPA 1999 datasets. Experimental results show that the proposed process detects up to 98.8% of false negatives and positives.

Deep learning algorithms are also proposed for the IDS, such as that in [55]. Compared to machine learning, deep learning was able to manage large volumes of data, and is thus most suitable for large datasets. In [41], due to extensive traffic from an IoT an anomaly-based IDS that uses a deep learning algorithm was applied. It began by taking packet features such as the IP address and reception rate into account. The intrusion detection process used a DBN with a feed-forward DNN. Hence, deep learning is a promising solution to identify attacks. In [56], the authors used preprocessing, feature extraction and classification phases. With preprocessing, the normalization process was applied and the significant features were extracted from the dataset. These features were then processed in the DBN layer, which is also used for signature verification. The process of matching the signatures of incoming packets with the database signature results is very tedious, plus it degrades attack detection accuracy and takes longer. An anomaly-based IDS model in which anomalous and normal packets are classified was proposed in [57]. Four different classifiers were used to detect the anomaly-based IDS: SVM, Decision Tree (DT), RF and Gradient Boosting Tree (GBT). The misused detection model operated using the Convolutional Long Short-term Memory (ConvLSTM) algorithm, with the features extracted by applying the convolutional algorithm then classified by the LSTM algorithm. The incoming packets are first processed in the anomaly-based IDS model, where high feature dimensionality of the incoming packet streams is not reduced. This causes high computation times and degrades the detection rate.

Table 2.3 shows a comparison of some of the previously mentioned works in the aspect of purpose, main features and some pitfalls for each. A brief comparison of related works in aspects of the dataset, IDS approach, algorithm and applications used are given in Table 2.4.

Table 2.4: Comparison of related works

Model	**Dataset**	**IDS Approach**	**Algorithms**	**Application**
Ensemble HIDS [46]	Bot-IoT	Signature and Anomaly-based IDS	C5 and SVM	IoT
Improving IDS by Estimating Parameters of RF in Boruta [47]	NSL-KDD dataset	Signature and Anomaly-based IDS	Random Forest (RF) and Boruta Algorithms	IoT
A Stacking Ensemble for Network IDS [48]	UNSW NB-15 and UGR'16	Signature-based IDS	Random Forest (RF), logistic regression, K Nearest Neighbor (KNN), and SVM	General Computer Networks
Hybrid IDS using K-means and Random Tree [49]	KDD'99 dataset	Signature-based IDS	K-means and Random Tree algorithms	General Computer Networks
Novel IDS based on an optimal hybrid kernel extreme learning machine [50]	UNSW-NB15 and KDD'99 datasets	Signature-based IDS	Gravitational Search Algorithm (GSA), Differential Evolution (DE), and Principal Component Analysis (KPCA) algorithms	General Computer Networks
A feature selection algorithm for IDS-based on PIO [51]	KDD'99, NLS-KDD and UNSW-NB15 datasets	Signature-based IDS	Pigeon Inspired Optimizer (PIO)	General Computer Networks
Anomaly network-based IDS using a reliable hybrid ABC and AdaBoost algorithms [52]	NSL-KDD and ISCXIDS2012 datasets	Anomaly-based IDS	Artificial Bee Colony (ABC) and AdaBoost algorithms	General Computer Networks
IDS for IoT Based on Improved Genetic Algorithm and Deep Belief Network [53]	NSL-KDD dataset	Signature-based IDS	Genetic Algorithm (GA) and Deep Belief Network (DBN) algorithms	IoT
Enhancing the Accuracy of IDSs by Reducing the Rates of False Positives and False Negatives Through Multi-objective Optimization [54]	DARPA 1999 and NSL-KDD datasets	Signature-based IDS	Multi-Objective Optimization Problem (BMOP)	General Computer Networks
Towards Deep-Learning-Driven IDS for the IoT [41]	Captured IoT network-traffic	Anomaly-based IDS	Deep Belief Network (DBN) and Deep Neural Network (DNN)	IoT
Deep Belief Network enhanced IDS to Prevent Security Breach in the IoT [56]	IoT network-traffic	Signature-based IDS	Deep Belief Network (DBN)	IoT
A Scalable and Hybrid IDS-Based on the Convolutional-LSTM Network [57]	ISCX-UNB dataset	Anomaly and misused-based IDS	Convolutional-LSTM Network	IoT

2.4 Survey on Transfer and Federated Learning (FL/TL) techniques to secure IoT applications

ML in general and DL primarily have been used to perform many different tasks in many IoT applications such as perception and predictions tasks, planning, decision making and control; a detailed surveys on the existing works on ML technologies for communications and networking systems can be found in [58],[59]. Securing those applications also has a good part in those tasks. In this section, we will discuss using the state-of-the-art ML/DL techniques, Federated and Transfer Learning (FL/TL), that have been proposed to secure the different IoT applications; mainly IoV and IoMT (See [60], [61]).

For the transfer learning technique, the authors in [62] were able to enhance the detection rate of the IoV intrusion detection system by 23% using the transfer learning technique, based on whether or not the IoV cloud model contributes partially to the labelled data to the connected vehicles during the model update process. Upon considering the two methods for the model updating, a cloud-assisted update scheme and a local model update scheme, the connected vehicles in the proposed model were able to detect the new attacks autonomously without the help of the cloud model. The work [63] designed a transfer learning-based IDS that used the Convolutional LSTM to detect new attacks by applying one-shot learning. The proposed model achieved 26.60% better performance than the other baseline machine learning algorithms using the CAN real traffic dataset. A model for detecting malicious attacks in the IoV environment is created using oversampling, outlier detection, and metric learning by the authors in [64]. The proposed approach used genetic algorithm to extract the optimal subset of features. The experimental results on the UNSW-NB15 dataset show that the proposed method achieved 98.51%, and 0.82% of accuracy and False Alarm Rate (FAR), respec-

tively. Transfer Learning has also been used with the help of deep learning to detect attacks in IoT networks. In their paper [65], Using two AutoEncoders (AEs), the authors present a technique for deep transfer learning (DTL). This technique allows users to learn from IoT data collected from multiple devices, not all of which are labelled. One of the two AutoEncoders (AEs) is used to train the source datasets (source domains) and another for training the target datasets (target domains). Experimental results of their model show better detection accuracy in detecting IoT attacks than other standard deep learning approaches and two recent methods. In [66], the authors proposed a model for distributing trust across multiple policy decision points. The authors' paper studied various threshold signature schemes. They identified an appropriate scheme for the policy decision points distribution to reduce the latency and increase the model performance as much as possible. As part of another work, [67], the authors developed a policy enforcement framework to overcome some of the current challenges of the network risk-based access control. They developed a policy language to support this framework, which includes a generic firewall rule language and a mechanism to map these rules to specific firewall syntax, so they can be implemented on a firewall. The authors in [68] have also considered transfer learning techniques for Natural Language Processing (NLP) by introducing a unified framework that converts all text-based language problems into a text-to-text format. Transfer learning has also been applied in computer vision. The authors in [69] have demonstrated the use of transfer learning in image classification and segmentation tasks. In [70], the authors have used deep transfer learning for classifying patients as COVID-infected or not from the COVID-19 chest CT scan images. The authors in [71] have considered transfer learning for hyperspectral image classification using a convolutional neural network and have concluded that transfer learning can be used between different hyperspectral images and be helpful to improve the classification

efficiency. The use of transfer learning in the NLP landscape can be seen in [72]. The authors present an overview of modern transfer learning methods in NLP tasks.

Federated learning technique has also been utilized to secure IoT applications; Zhao *et al.* in [73] have used federated learning aided long short-term memory (LSTM) framework for intrusion detection. The authors have shown that their proposed scheme achieves higher accuracy and better consistency than conventional approaches. The authors in [74] have considered federated learning for privacy-preserving intrusion detection. Li *et al.* in [75] have proposed a novel federated deep learning scheme, called DeepFed, in detecting various types of cyber threats against industrial cyber-physical systems (CPSs). Sun *et al.* in [76] have proposed a segmented federated learning approach for large-scale multiple LANs. Their proposed scheme allows each segment of participants to conduct collaborative learning separately and rearranges the segmentation of participants dynamically. The authors in [77] have considered a multi-task anomaly detection approach using federated learning in which network anomaly detection task, traffic recognition task, and traffic classification task are tackled simultaneously. Mothukari *et al.* in [78] have proposed an anomaly detection approach based on federated learning to proactively recognize intrusion in IoT networks. The authors in [79] have used federated learning with blockchain technology that supports the auditing of machine learning models without the necessity to centralize the training data in an intrusion detection case study. Yang *et al.* in [80] have listed several applications, challenges and opportunities of federated learning in the 6G technology domain. The authors in [81] have developed a testbed that utilizes the ML capability for managing security issues using a variety of healthcare sensors. The testbed includes a gateway for data gathering, an IDS computer for monitoring the network traffic and detecting abnormal behaviors, an attacker to imitate a real attack threat to the system, and a server. In [82], the authors proposed a decentral-

ized, federated Blockchain-based framework for vehicular networks, in which the private information for the emergency vehicles is guaranteed by adopting the federated learning with the blockchain capability in the fog side. Their scheme gives good results in terms of latency, throughput, accuracy rate, lifetime reduction and energy consumption. A brief survey of existing studies of federated learning for vehicular IoT can be found in [83].

In [84], the authors designed an IDS for the ICV environment to detect three basic attacks, namely, DoS, spoofing, and sniffing attacks and used a tree-structure machine learning algorithms and used a mixture of SMOTE oversampling algorithms and tree-based averaging feature selection methods. The evaluation in that work shows very good detection accuracy and a 38.6% decrease in the computational time when compared with other similar methods using the CICIDS2017 dataset. However, the model design does not consider the attack signatures and only works as an anomaly-based IDS, which will increase the monitoring overhead for resource-constrained vehicles. In [85], an IDS has been proposed to secure in-vehicle networks to detect message injection attacks based on the analysis of time intervals of CAN messages. The proposed scheme is able to reduce the detection delay and achieves high detection accuracy in different types of message injection experiments. The authors in [86], designed an IDS that can detect spoofing and jamming attacks in an ICV network. The model used different machine learning algorithms such as RF, k-Nearest Neighbour (k-NN) and One-Class Support Vector Machine (OCSVM) to detect known and unknown attacks with 90% detection accuracy. An IDS based on probabilistic cross-layer to detect spoofing attacks is introduced in [87]. Their model used the two well-known machine learning algorithms, k-NN and RF classification, and achieved 91% of accuracy in each of them. In [88], the authors proposed an IDS to secure CAN communication protocol that is monitors the offset ratio and time interval

between request and response messages to detect the attacks in the connected vehicles. This method is based on the fact that these parameters are fixed within the normal traffic and vary when attacks happen. The performance evaluation for their scheme shows fast response and high accuracy.

2.5 Summary

This chapter reviewed significant ML and DL approaches to achieving secure IoT environments in this chapter. We presented a brief overview of IoT architecture, IoT security issues, Transfer learning, Federated Learning, and ML/DL approaches used for IoT security. Then discussed and compared state-of-the-art ML and DL algorithms such as CNN, RNN, LSTM, AE, RBM, DBN and DRL. We also evaluated crucial recent work in the IoT security field, explained the role of various ML and DL techniques and identified significant problems with the state-of-the-art approaches. Our extensive examination highlighted research problems and gaps in current IoT security schemes. Finally, we proposed promising future directions for ML/DL-based security in IoT environments.

Chapter 3

Deep Learning-based IDS for IoT

3.1 Introduction

Machine Learning (ML) and Deep Learning (DL) methods have recently been proposed to detect and mitigate security threats [89, 90]. For attack detection, ML algorithms such as Support Vector Machines (SVM), K-Nearest Neighbour algorithm (KNN), Decision Trees (DT), Bayesian algorithms, Random Forest and K-means algorithm are typically applied in IDS. However, since traditional methods usually lack optimal features learning and dataset management, the accuracy of attack detection cannot be guaranteed. Dealing with irrelevant features in high dimensional data generated from thousands of IoT sensors and devices can increase the potential for over-fitting, making decisions based on noise and extending training time. Similarly, DL approaches such as Deep Belief Network (DBN), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Restricted Boltzmann Machine (RBM) and deep AutoEncoder (AE) are also used in IDSs. However, the DL approach has achieved better performance than ML, particularly in large datasets. Several IoT systems produce substantial volumes of data, and DL methods are suitable for these systems since they support high-dimensional features.

They also enable deep linking, which allows IoT-based devices and their applications to interact with one another without human intervention. Though security and privacy are the primary goals of DL IoT data processing, it has also been widely used for image processing, Big Data analytics, video processing and speech processing in IoT and smart city applications. Therefore, to address the challenges of protecting the IoT, we propose a novel, deep learning-based framework for IoT environments. In the proposed model, we will show how Spider Monkey Optimization (SMO) benefits our system by choosing the optimal features in Section 3.2.2.

3.2 Deep Learning-based IDS model (DL-IDS)

This section provides detailed explanations about using DL-IDS [91] to secure IoT environments. The overall process of DL-IDS is depicted in Figure 3.1, and shows that it begins with preprocessing to remove uncertainties in the normalized dataset. Preprocessing applies two effective processes: redundancy elimination and missing value replacement. The cleaned dataset is then processed with an optimal feature selection algorithm to extract relevant features, and based on these the data is classified into normal and anomalous data.

Here, we consider four categories of anomalies (DoS, Probing, U2R and R2L) which can be defined as follows:

1. DoS: An attacker tries to make a service unavailable to legitimate users by uploading enormous unwanted packets. DoS attacks include Apache2, Back, Land, Udpstorm and Smurf.

2. Probing: In this type of attack, an attacker monitors the remote victim and tries to collect some information by doing the port scanning. Attacks such as duration

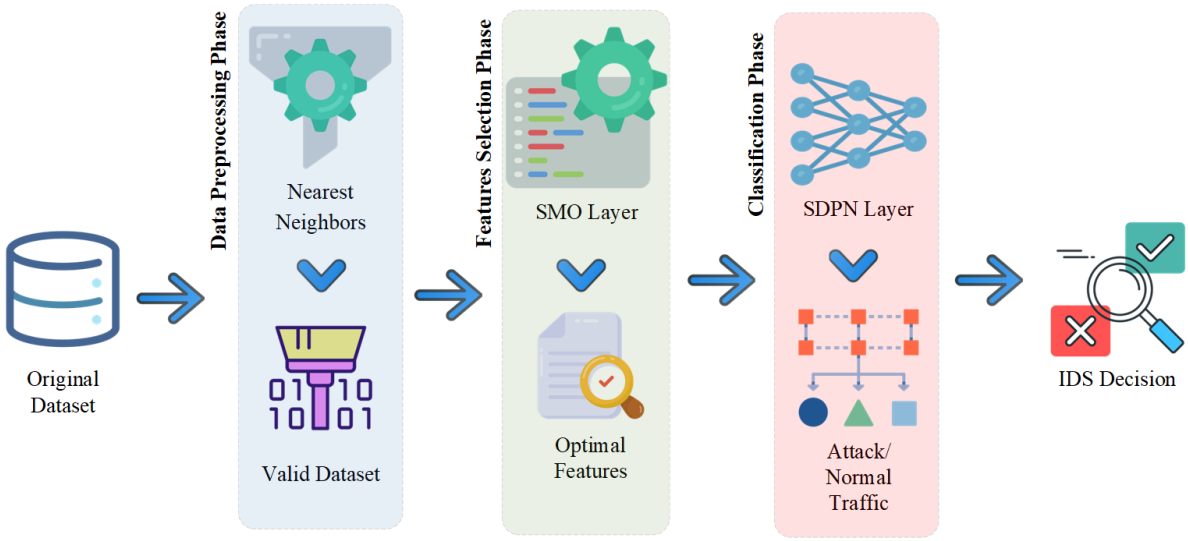

Figure 3.1: DL-IDS Framework

of the connection and source bytes are some types of Probing attack.

3. R2L: In this type of attack, an unauthorized attacker attempts to access the system without a system account. R2L attacks include Ftpwrite, Guesspasswd and SNMP.

4. U2R: In this scenario, the attacker has local access to a victim's machine and tries to obtain legitimate user privileges. Buffer overflow, HTTP tunnel and rootkit attacks are types of U2R attacks.

3.2.1 Preprocessing

This process is initialized with similarity measurements of the data in the dataset, using the Minkowski distance to compute the distance between each pair of data. Duplicate and redundant data are then removed from the dataset and fed into the next stage, where the missing values are replaced by nearest neighbour computation to avoid the classifier to biased towards more frequent records. Missing value replacement, or imputation, is the process of predicting the relevant value of missed attributes in the data. This missing

value replacement method identifies the nearest neighbours of missing values, and the mean value of the missing attributes is replaced in the missing values.

Consider a dataset with m records in which each record has n attributes, and x attributes are missing in the data record R_1. Then, the K nearest neighbours of R_1 are determined according to the Euclidean Distance (ED) as follows:

$$ED = \sum_{i=2}^{m} |R_1 - R_i|^2 \tag{3.1}$$

Based on the distance value, the K nearest neighbour records are identified for R_1, and the attribute x value of R_1 is computed from the attribute value of its K neighbours. The mean value of x is computed for the x values presented in neighbour records, and the missing value is replaced by this mean value. This step eliminates all redundant data, and the missing values are replaced to remove all uncertainties from the dataset.

3.2.2 Optimal feature selection using Spider Monkey Optimization (SMO)

In the preprocessing step all uncertainties are removed from the dataset, and in this step we select the optimal features from the dataset. Optimal feature selection minimizes feature learning time for our SDPN classifier. For optimal feature selection [92] we adapt the SMO algorithm, which is a food foraging-based optimization algorithm inspired by fission-fusion social systems. The algorithm has been used in many engineering domains since it requires fewer control parameters, which makes it easier to apply in complex optimization problems. The following are the main control parameters with their values:

- Local Leader Limit = [S x D]
- Global Leader Limit = [S/2, S x 2]

- Maximum no. of groups MG = [S/10]
- Perturbation rate, PR = [0.1, 0.9]

where S is the population size and D is the dimension. Algorithm 1 explains the process of optimal feature selection with the SMO algorithm. The details of the SMO steps follow:

Algorithm 1: SMO based feature selection

```
Begin
Set value of control parameters
Initialize swarm with initial population of SMOs
Compute fitness for all SMO in the swarm
Select global leader and local leader
Set iteration = 0 while (Iteration < Maximum iterations) do
    //Local Leader Phase;
    Compute the probability of a particular solution
    Update the position of each SMO in the group
    //Global Leader Phase;
    Update the position of each SMO in the swarm
    //Global Leader Learning phase;
    Update the position of the global leader
    Select global leader with highest fitness
    //Local Leader Learning phase;
    Update position of the local leader
    Select local leader based on fitness
    Iteration = Iteration+1;
end while
End
```

Initialization: At this step, all solutions are initialized as spider monkeys. In our work, the 41 features are initialized as a population of N spider monkeys, where each monkey $SM_i (i = 1, 2, ..., N)$ is a D-dimensional vector, and SM_i indicates the i^{th} Spider Monkey (SM) in this population. Each dimension j of SM_i is initialized as follows:

$$SM_{ij} = SM_{minj} + Rand\,[0, 1]\,(SM_{maxj} - SM_{minj}) \tag{3.2}$$

where, SM_{minj} and SM_{maxj} are limits of SM_{ij} in the j^{th} direction, and $rand[0,1]$ is a random value in the range of 0 to 1.

Local Leader Phase (LLP): In this stage, each SM modernizes its present location based on the experience of every local leader and local group members, and the fitness value of the new position is defined. If the fitness value of the new position is better than the current, with higher being better, the spider monkey updates its position with the new one, as follows:

$$SM_{newij} = SM_{ij} + rand\,[0,1]\,(LL_{kj} - SM_{ij}) + rand[-1,1](SM_{rj} - SM_{ij}) \tag{3.3}$$

where the j^{th} dimension of the i^{th} SM is indicated by SM_{ij}, the j^{th} dimension of the k^{th} local group leader position is denoted by LL_{kj}. SM_{rj} indicates the j^{th} dimension of the r^{th} SM which is selected randomly from the K^{th} local group where r $\neq i$.

Global Leader Phase (GLP): After concluding the local leader phase, all spider monkeys can update their positions based on the experience of the global leader and local group members, as follows:

$$SM_{newij} = SM_{ij} + rand\,[0,1]\,(GL_{j} - SM_{ij}) + rand[-1,1](SM_{rj} - SM_{ij}) \tag{3.4}$$

where, GL_j is the j^{th} dimension of the GLP and $j \in \{1, 2, \ldots, D\}$ is a randomly selected index. The positions of the spider monkeys (SM_i) are updated based on their probability $Prob_i$, which is defined by the means of their fitness. In this way, the best candidate will have higher probability to be the leader in the next phase. The probability $Prob_i$ is calculated as follows (or by any other function of fitness):

$$Prob_i = \frac{Fitness_i}{\sum_{i=1}^{N} Fitness_i} \tag{3.5}$$

where $Fitness_i$ refers to the fitness value of the i^{th} SM. The fitness of the last position is defined and compared to the previous one, and the best is then selected. In this work, the evaluation of each feature (spider monkey) is analyzed by an SDPN classifier in order to calculate the efficiency of the signified feature subset (i.e. the accuracy).

Global Leader Learning phase (GLL): In this phase, the global leader modifies its location by applying a greedy global selection to all the population sets, and the SM with the highest fitness value becomes the global leader. A counter *GlobalLimitCount* is incremented by 1, in case the global leader position does not update.

Local Leader Learning phase (LLL): In this phase, the local leader in each group updates its position by applying the greedy selection of its group, and the SM with the highest fitness value in each group becomes the local leader. A counter *LocalLimitCount* is incremented by 1 in case the local leader position does not update.

Local Leader Decision (LLD): In this phase, if the Local Limit Count reaches its threshold value the group members change their positions by using information from the global and local leaders by the equation:

$$SM_{newij} = SM_{ij} + U(0,1) * (GL_j - SM_{ij}) + U(0,1) * (SM_{ij} - LL_{kj}) \tag{3.6}$$

Global Leader Decision (GLD): In this phase, if the global leader does not reach the maximum number of iterations (i.e. Global Leader Limit), the leader divides the population into smaller local groups until they reach the maximum number of groups (MG). In this case, if the position of global leader is not updated it combines all the groups into one.

The best subset of features with high classification accuracy becomes the optimal result. Thus, the algorithm simulates the fusion-fission social system of SMs, and in the process the optimal feature set is selected by the SMO algorithm. Table 3.1 shows the NSL-KDD

features selected by SMO that give the best accuracy. SMO helps reach the highest accuracy values (as will be discussed in the next Section), by choosing a combination of optimal features that can achieve these values (i.e. 20 features out of a full features set). However, the best accuracy percentage we had obtained without SMO was 96%.

Table 3.1: Selected Features by SMO

No.	**Selected By SMO**	**Type**	**Feature Name**	**No.**	**Selected By SMO**	**Type**	**Feature Name**
1		Cont.	duration	23	√	Cont.	count
2	√	Symb.	protocol_type	24		Cont.	srv_count
3	√	Symb.	service	25	√	Cont.	serror_rate
4	√	Symb.	flag	26		Cont	srv_serror_rate
5	√	Cont.	src_bytes	27		Cont.	rerror_rate
6	√	Cont.	dst_bytes	28		Cont.	srv_rerror_rate
7	√	Symb.	land	29	√	Cont.	same_srv_rate
8	√	Cont.	wrong_fragment	30	√	Cont.	diff_srv_rate
9		Cont.	urgent	31		Cont.	srv_diff_h_rate
10	√	Cont.	hot	32		Cont.	dst_h_count
11		Cont.	num_failed_logins	33		Cont.	dst_h_srv_count
12	√	Symb.	logged_in	34		Cont.	dst_h_same_srv_rate
13		Cont.	num_compromised	35	√	Cont.	dst_h_diff_srv_rate
14		Cont.	root_shell	36	√	Cont.	dst_h_same_src_port_rate
15		Cont.	su_attempted	37	√	Cont.	dst_h_srv_diff_h_rate
16		Cont.	num_root	38	√	Cont.	dst_h_serror_rate
17		Cont.	num_file_creations	39	√	Cont.	dst_h_srv_serror_rate
18		Cont.	num_shells	40	√	Cont.	dst_h_rerror_rate
19		Cont.	num_access_files	41		Cont.	dst_h_srv_rerror_rate
20		Cont.	num_outbound_cmds				
21		Symb.	is_h_login				
22		Symb.	is_guest_login				

Cont.=Continuous
Symb.=Symbolic

3.2.3 Stacked Deep Polynomial Network (SDPN)

In the classification phase, we adopted a Stacked Deep Polynomial Network (SDPN) that has been used with numerous small and large datasets that have shown high-performance results in [93, 94]. In SDPN, multiple basic DPNs are stacked to form a deep hierarchy in order to develop a new, supervised deep neural network algorithm with efficient layer-by-layer learning. The output of each node is a quadratic function of its inputs, and the highest-level output representation is then fed to an SDPN classifier. The elimination of irrelevant features by selecting optimal features minimizes SDPN training time, and the optimal technique of building deep networks in SDPN allows data representation learning on small finite samples. The algorithm starts with a simple network that could have significant bias but will not tend to over-fit (i.e. low variance), and as the network becomes deeper, the bias gradually decreases while increasing the variance. Thus, this algorithm could be applied to train the natural curve of the solutions used to control the bias-variance trade-off. Construction of the layers in DPN begins by assigning the degree-1 polynomial function (linear) over the training dataset, where the bias is built using Singular Vector Decomposition (SVD) to generate an independent set of vectors. The outputs of the first layer extend all the values obtained by the linear functions to the training instances. Then, using the same concept, the basis for degree-2 and three polynomials can be derived. This means that the vector of values achieved by any degree-2 polynomial extends the vector of values obtained by nodes in the first layer and the products of the outputs of every two nodes in the first layer.

Letting $R = \{r_1, r_2, ..., r_m\} \in R^{mxd}$ gives a set of m training samples, where r_i is a d-dimensional sample. Then, the construction of the first layer in DPN begins by assigning the degree-1 polynomial function (linear) over the training data as follows:

$$\{ (\langle w \ , \ [1 \ \ r_1] \rangle \ , \ ... \ , \ \langle w \ , \ [1 \ \ r_m] \rangle) : \ w \ \in \ R^{d+1} \} \tag{3.7}$$

which is the (d+1)-dimensional linear subspace of R_m. Therefore, to build the base we need to find d+1 vectors $w_1, \ldots, w_{d+1}$. So, $\{ (\langle w_j \ , \ [1 \ \ r_1] \rangle \ , \ ... \ , \ \langle w_j \ , \ [1 \ \ r_m] \rangle) \}_{j=1}^{d+1}$ are linearly independent vectors. This can be generated using Singular Vector Decomposition (SVD) to achieve linear transformation that is specified by matrix W, and maps [1 X] into the generated base where the columns of W specify the d+1 linear function that form the first layer of the network. The j^{th} node of the first layer functions is:

$$n_j^1 \ (r) \ = \ \langle W_j \ , \ [1 \ R] \rangle \tag{3.8}$$

where $\{ \ (\ n_j^1(r_1) \ , \ ... \ , \ n_j^1(r_m) \ ) \ \}_{j=1}^{d+1}$ is the basis for all values obtained by degree-1 polynomials over the training dataset. If F^1 refers to the $m \times$ *(d+1)* matrix in which columns are the vectors of the matrix, then $F_{i.j}^1 = n_j^1(r_i)$.

This is how the network of the first layer, whose outputs extend all the values obtained by linear functions on the training dataset, is constructed. The basis of degree-2 and 3 can be determined in the same way, then any degree s polynomial can be expressed as:

$$\sum_{\mathrm{i}} \mathrm{g_i(r)} \ \mathrm{h_i(r)+k(r)} \tag{3.9}$$

where, $g_i(r)$ are degree-1 polynomials, $h_i(r)$ are $(s-1)$-degree polynomials, and $k(r)$ is the polynomial of degree at most $s-1$. The nodes in layer-1 first extend all degree-1 polynomials, then extend polynomials g_i, h_i and k. Thus, we can express any degree-2

polynomial as:

$$\sum_i \left(\sum_j {\alpha_j}^{(g_i)} n_j^1(r) \right) \left(\sum_x {\alpha_x}^{(h_i)} n_x^1(r) \right) + \left(\sum_j {\alpha_j}^{(k)} n_j^1(r) \right) = \left(\sum_{j,x} n_j^1(r) n_x^1(r) \right) \left(\sum_i {\alpha_j}^{(g_i)} \alpha_x^{(h_i)} \right) + \sum_j n_j^1(r) (\alpha_j^{(k)}) \tag{3.10}$$

where α's are scalar factors. This means the vector of values attained by any degree-2 polynomial extend both the vector of values from nodes layer and the products of outputs of every two nodes in the first layer. For algebraic representation, when the first layer was constructed a matrix F^1 was formed, in which columns extend all values obtained by degree-1 polynomials. Then the matrix [F $\widetilde{F}^2$] can be expressed as:

$$\widetilde{F}^2 = [(F_1^{1\circ} \ F_1^1) \ldots (F_1^{1\circ} \ F_{|F_1|}^1) \ \ldots (F_{|F^1|}^{1\circ} \ F_1^1) \ldots (F_{|F^1|}^{1\circ} \ F_{|F1|}^1)] \tag{3.11}$$

where, $^\circ$ refers to the direct matrix product. We can define F^2 as the method as F^1, then the new matrix$[F \ \ \widetilde{F}^2]$ extends all possible values of degree-2 polynomials, then $[F \ F^2]$'s columns are linearly independent basis for $[F \ \widetilde{F}^2]$'s columns. To construct layer 3, we simply repeat the previous process at each iteration s to maintain matrix F, whose columns form a basis for the values obtained by all polynomials of degree $\leq s-1$. We can then write the new matrix as:

$$\widetilde{F}^s = [(F_1^{s-1\circ} \ F_1^1) \ldots (F_1^{s-1\circ} \ F_{|F_1|}^1) \ldots (F_{|F^{s-1}|}^{s-1} \ F_1^1) \ldots (F_{|F^{s-1}|}^{s-1} \ F_{|F1|}^1)] \tag{3.12}$$

After some $\Delta - 1$ iterations, a matrix F is constructed whose columns form a basis for all values obtained by polynomials of degree $\leq \ \Delta - 1$ over the training dataset. Algorithm 2 summarizes the steps for building DPN.

To verify that the model has high performance with both training and test data

Algorithm 2: Main algorithm flowchart to build the networks in DPN

Begin
- Initialize an empty matrix (F) and $\widetilde{F}^{\,1} := [1 \;\; \mathrm{X}]$
- Find W by computing SVD of $\widetilde{F}^{\,1}$
- Build basis for layer 1: $(\mathrm{F}^{\,1}, \mathrm{W}^{\,1}) := (\widetilde{F}^{\,1})$
- // Columns of F^1 are linearly independent, and $\mathrm{F}^1 := \widetilde{F}^{\,1}\ \mathrm{W}^1$
- Building first-layer: $\forall i \in \{1, \ldots, |F^1|\}, n_i^1(r) := \langle W_i, [1\ R] \rangle$
- $F := F^1$
- Build higher layers s = 2 and 3:
- $\widetilde{F}^{\,s} := [\,(F_1^{s-1} \circ F_1^1) \ldots (F_1^{s-1} \circ F_2^1) \ldots (F_{|F^{s-1}|}^{s-1} \circ F_{|F^1|}^1)\,]$
- Build basis for layer s: $(\mathrm{F}^{\,s}, \mathrm{W}^{\,s}) := (\widetilde{F}^{\,s})$
- // Columns of $[\mathrm{F}\ \mathrm{F}^1]$ are linearly independent
- $\mathrm{F} := [\mathrm{F}\ \mathrm{F}^s]$

End

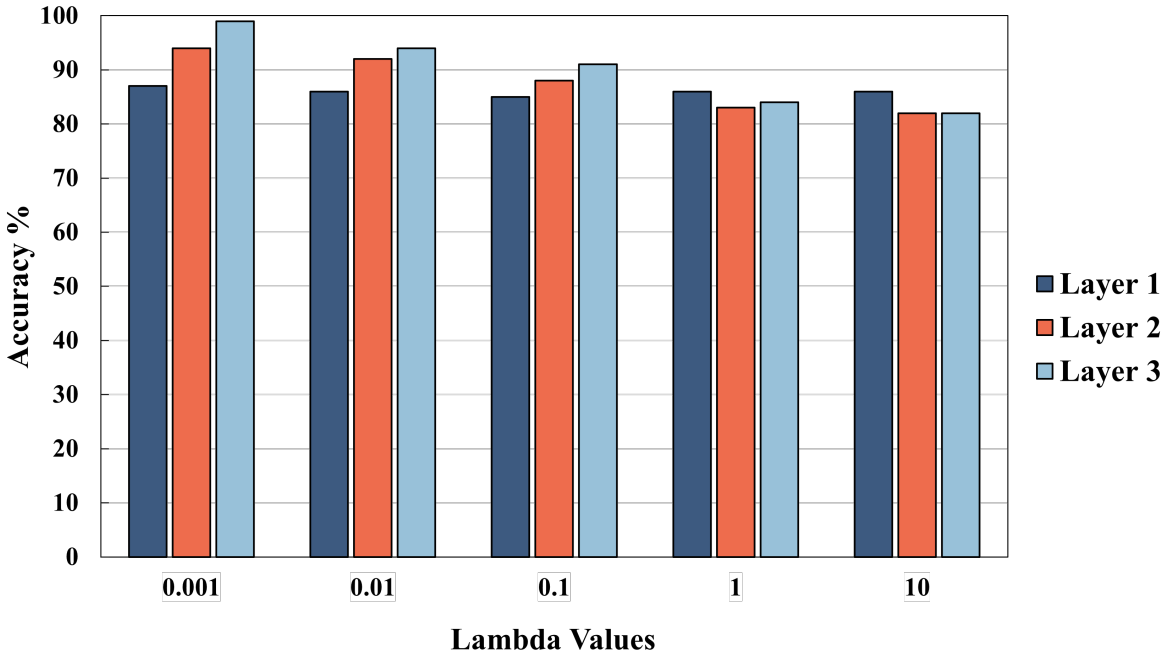

Figure 3.2: Accuraccy related to SDPN layers with different lambda values

(i.e. avoids over-fitting), we use the L2 regularization technique. This extends the loss function by adding a regularization term with a regularization parameter known as a lambda value (λ) which controls how to manage the features. It is a hyperparameter with no fixed value, and when its value increases the features are highly penalized and, as a result, the model becomes simpler. When its values decrease there is minimal penalization of the features, and the model becomes more complex. Figure 3.2 shows the accuracy related to the SDPN layers using different values of lambda (λ). This accuracy refers to the percentage of correct predictions and shows that each layer has a different level of accuracy. Through our experiments, we determined that the highest accuracy is achieved in layer 3 with a lambda value of 0.001. However, since the SDPN algorithm is not necessary to specify the number of iterations, we can simply create the layers one-by-one, check the performance of the resulting network on a validation set, and stop once we reach satisfactory performance.

With this methodology, all optimal features selected by SMO algorithms are learned in each layer of SDPN, and high-level and optimal features are classified. Building on these important features, SDPN classifies the data into normal and anomalous, based on the kernel function. Our proposed DL-IDS system performs preprocessing, feature selection and classification for intrusion detection, and the elimination of uncertainties in the dataset improves attack detection performance. Optimal feature selection improves the accuracy of the classifier, and SDPN maps the optimal low-level features to high-level features.

Figures 3.3, 3.4, and 3.5 show comparisons of the time cost to build SDPN layers with different values of lambda (λ) before and after using SMO, and highlights the time saved using the SMO optimal features selection. The SMO reduced training time an average of 66.59% when the dimension of the dataset was reduced (i.e. from 41 features to 20

features). This also increased the classification speed, as well as decreasing calculations that can lower the algorithm running time which indicates that this model can be used in the real time environments.

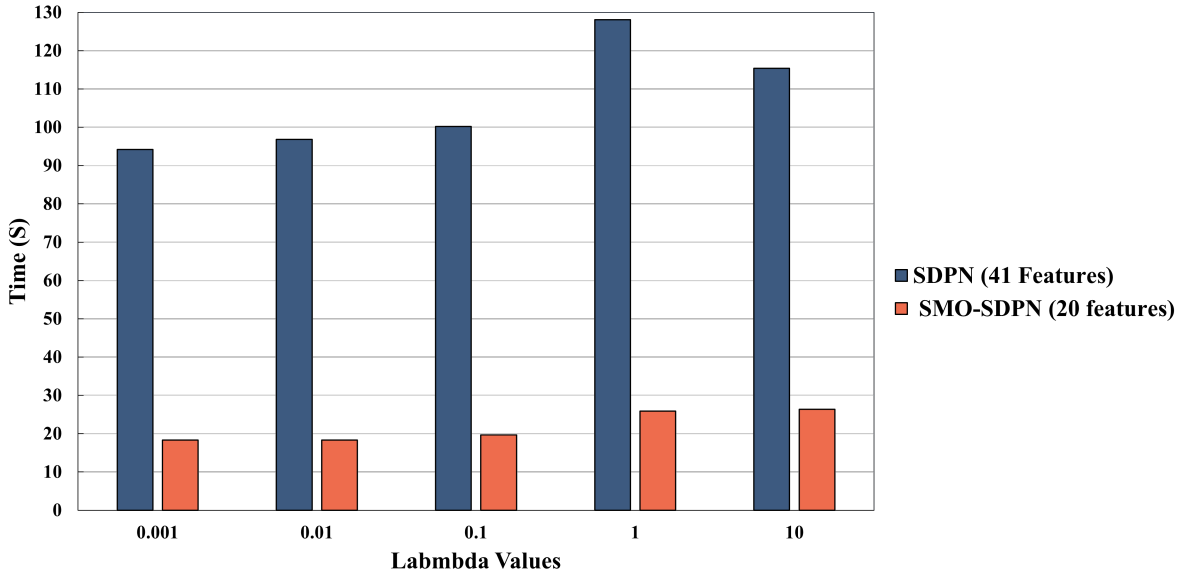

Figure 3.3: Comparison of Time Cost to Build the SDPN Layer 1

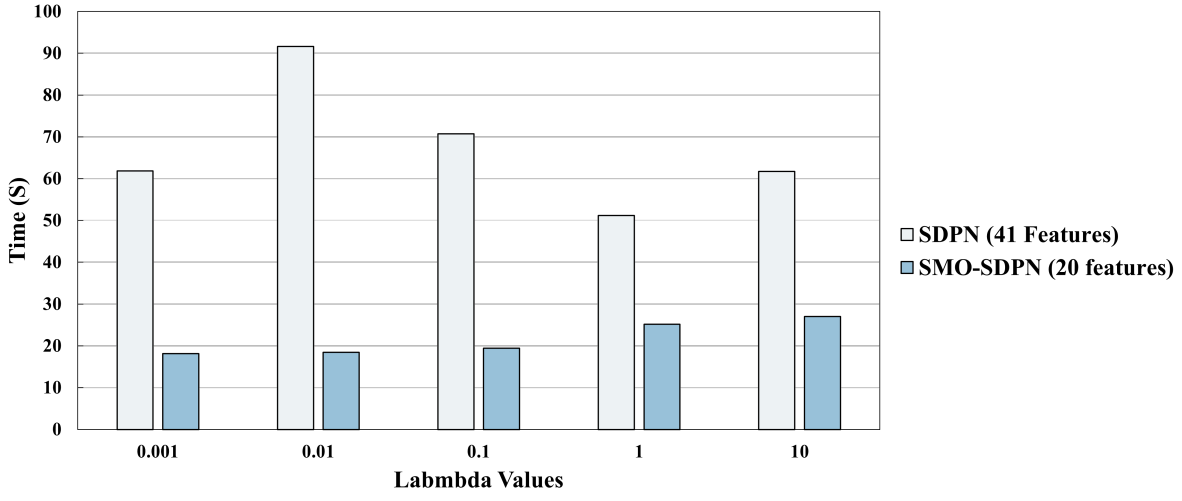

Figure 3.4: Comparison of Time Cost to Build the SDPN Layer 2

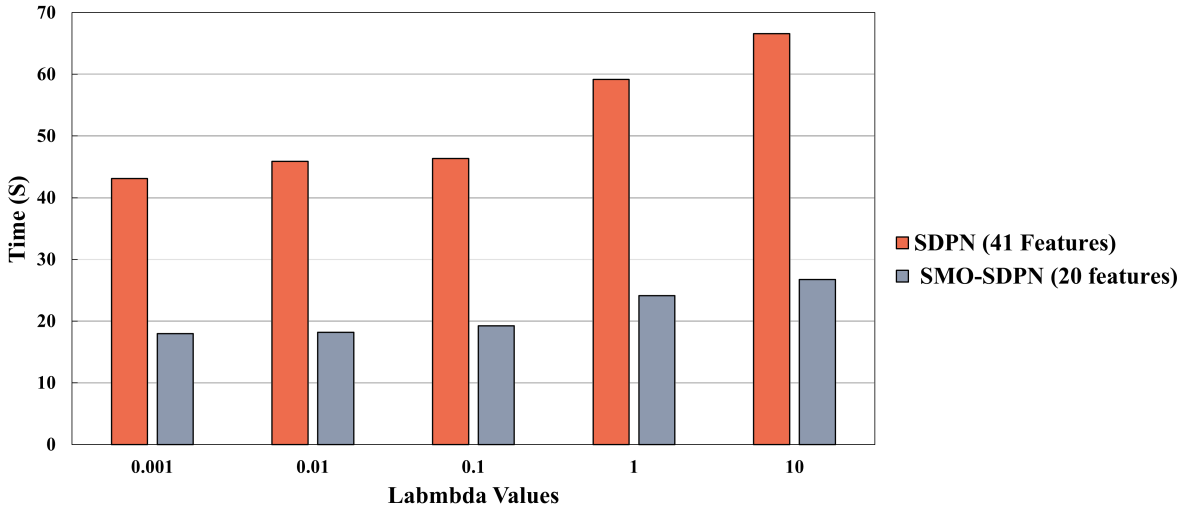

Figure 3.5: Comparison of Time Cost to Build the SDPN Layer 3

3.3 Performance Evaluation

In this section, we analyze the performance of the proposed DL-IDS system, based on significant performance metrics, such as accuracy, precision, recall and F-score.

3.3.1 NSL-KDD Dataset

To evaluate the performance of DL-IDS, we used the NSL-KDD benchmark dataset, which is the most recent version of the KDD'99 dataset [95]. This dataset contains DoS, R2L, probe and U2R Cyber-attacks, and consists of 125,973 records for training and 22,554 records for testing. It involves 41 features, including duration, protocol, service, flag, source bytes and destination bytes. Each NSL-KDD record has 41 features (e.g., *protocoltype*, $Logged_{in}$, and *Duration*). These features are represented as numeric, nominal, and binary,defined as continuous or discrete, and labelled as normal or attack. The training and testing sets contain a total of 22 and 17 attack types, respectively. A brief analysis of the NSL-KDD dataset that shows the number of attack records in the training and testing sets is provided in Table 3.2, and Table 3.3 describes each of

NSL-KDD attacks types.

Table 3.2: Dataset analysis

Traffic Type		***Training***	***Testing***
Normal		67343	9711
Attacks	**DoS**	45927	7458
	Probe	11656	2754
	R2L	995	2421
	U2R	52	200
Total		**125973**	**22544**

Table 3.3: Attacks in NSL-KDD dataset

Attack Type	**Description**
DoS	It is a type of flooding data packets that occupies larger resource in the network.
Probe	This attack is defined based on the information gathering i.e. it collects data from the other nodes.
U2R	This attack defines the involvement of access requests from unauthorized root node or super user.
R2L	This attack performs on local access of unauthorized nodes using devices present in remote locations.

In DL-IDS, the dataset first undergoes the preprocessing phase in which redundant data are eliminated and missing values are replaced. Then optimal features are selected by the SMO algorithm, and further feature learning and classification with optimal features is performed by SDPN. We used Python-Tensorflow on CPU core i7 based machine to simulate our proposed framework, and for optimal feature selection 41 populations were initialized in the SMO algorithm, and 100 iterations were performed.

3.3.2 Performance Metrics

In this work, we considered for comparison performance metrics such as accuracy, precision, recall and F score, which are defined as follows:

Accuracy: This is defined as the percentage of correct predictions; that is, the percentage of anomalous traffic that is classified correctly. It is the ratio of correct detections to the total number of records in the dataset, and can be computed as follows,

$$Accuracy = (TP + TN)/(TP + TN + FP + FN) \tag{3.13}$$

Accuracy is computed based on false positive (FP) and true positive (TP) values.

Precision: This metric describes the classifier's ability to predict normal data without conditions. It is defined as the number of true positives divided by the number of true positives, plus the number of false positives as follows:

$$Precision = TP/(TP + FP) \tag{3.14}$$

Recall: Recall is the ratio of the number of records correctly classified to the number of all corrected events, and can be computed as follows:

$$Recall = TP/(TP + FN) \tag{3.15}$$

Here, FN represents the false negative.

F1-Score: This is defined as the harmonic mean of recall and precision, which can be computed as follows:

$$F1\text{-}Score = 2TP/(2TP + FP + FN) \tag{3.16}$$

All these metrics are significant for evaluation of the classifiers.

3.3.3 Comparative Analysis

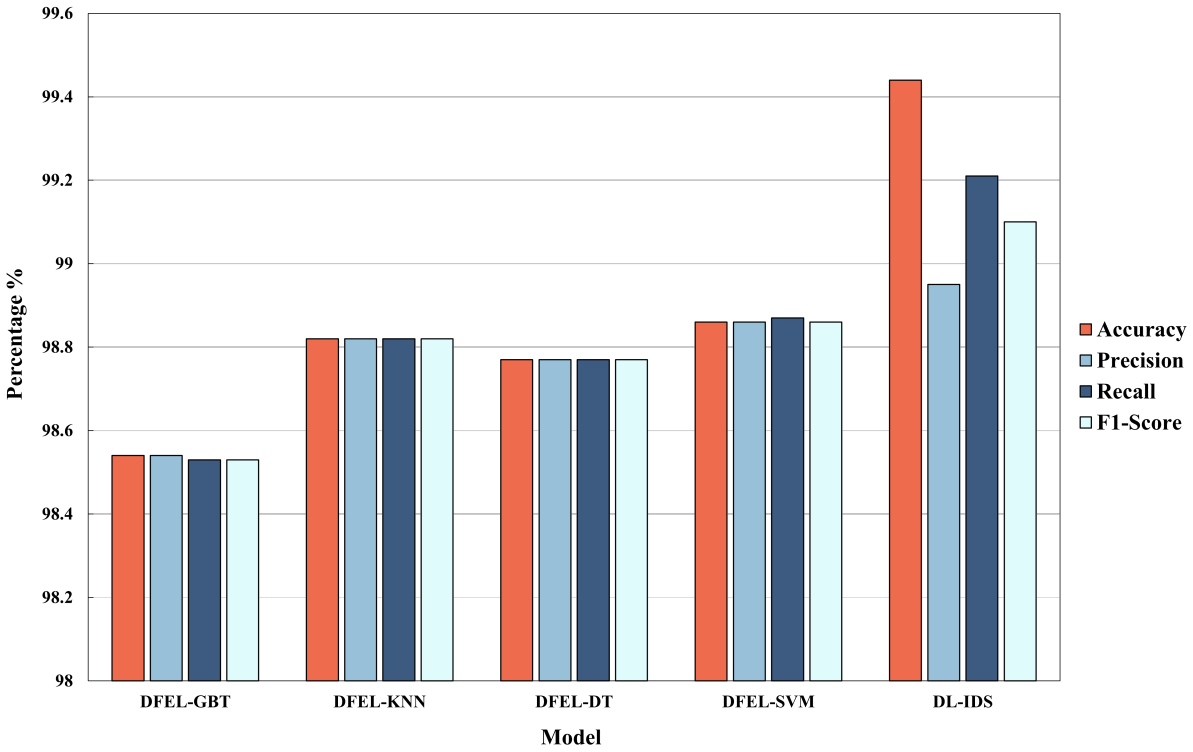

Figure 3.6: DFEL Models With DL-IDS Performance Comparison

Figure 3.6 shows comparisons of four Deep Feature Embedding Learning (DFEL) classifiers: K-Nearest Neighbours (KNN), Decision Tree (DT) and Support Vector Machine (SVM). DFEL architecture has been proposed for Intrusion Detection in IoT environments[96]. The most important performance evaluation metrics for attack detection include accuracy, precision, recall and F1 score, and these are used here to compare our model with DFEL versions. Accuracy, as expressed in Equation 3.13, is an evaluation the ability of the classifiers to correctly detect the intrusions. DFEL is based on different machine learning classifiers, and was intended to reduce the data dimensions through the use of edge-of-deep and transfer learning. Though it minimizes training time it does not improve the detection accuracy of the three classifiers, since our DL-IDS system has

an effective preprocessing process and optimal feature selections. Another metric that is significantly improved is precision, as shown in the figure. Precision is important as it expresses the rate at which False Positives (FP) will occur relative to the number of True Positives (TPs). Traffic classified as malicious requires an analyst's investigation, and a high rate of precision results in fewer FPs per alert for a security analyst to manage. The ideal rate of precision is 1, which means every alert shown to an analyst will be a TP. This is because DFEL is unable to predict the class of the data since it is designed to classify network traffic as normal or abnormal only. It also cannot deal with uncertain datasets, which is why DFEL models have two classifiers (i.e. DFEL-KNN and DFEL-DT) with the same performance metrics. This indicates that the accuracy is equal to the sensitivity of a recall, or True Positive Rate (TPR), and equal to the selectivity, or True Negative Rate (TNR). This means that the model is balanced and can simultaneously classify positive instances and negative instances correctly. Though sensitivity and specificity are important in IDS cases, being in balance is not necessarily positive. However, our model has higher precision than recall, which means that the model returned substantially more relevant results than irrelevant ones.

Similarly, we compared our model with the deep learning-based attack detection mechanism (D-DL) [31], the results of which are shown in Figure 3.7. In D-DL, a deep learning-based Cyber-attack detection mechanism for IoT was distributed over fog nodes using the NSL-KDD dataset. It was concluded that though the DL approach outperforms shallow learning methods, the detection efficiency is limited and requires further analysis with respect to payload-based detection. Based on the calculated average values of each performance metric of the deep model with the 5-classes (i.e. Normal, DoS, Probe, R2L and U2R), it is evident that DL-IDS achieves better accuracy, precision, recall and F1-Score performance, due to D-DL being processed with non-optimal features. The

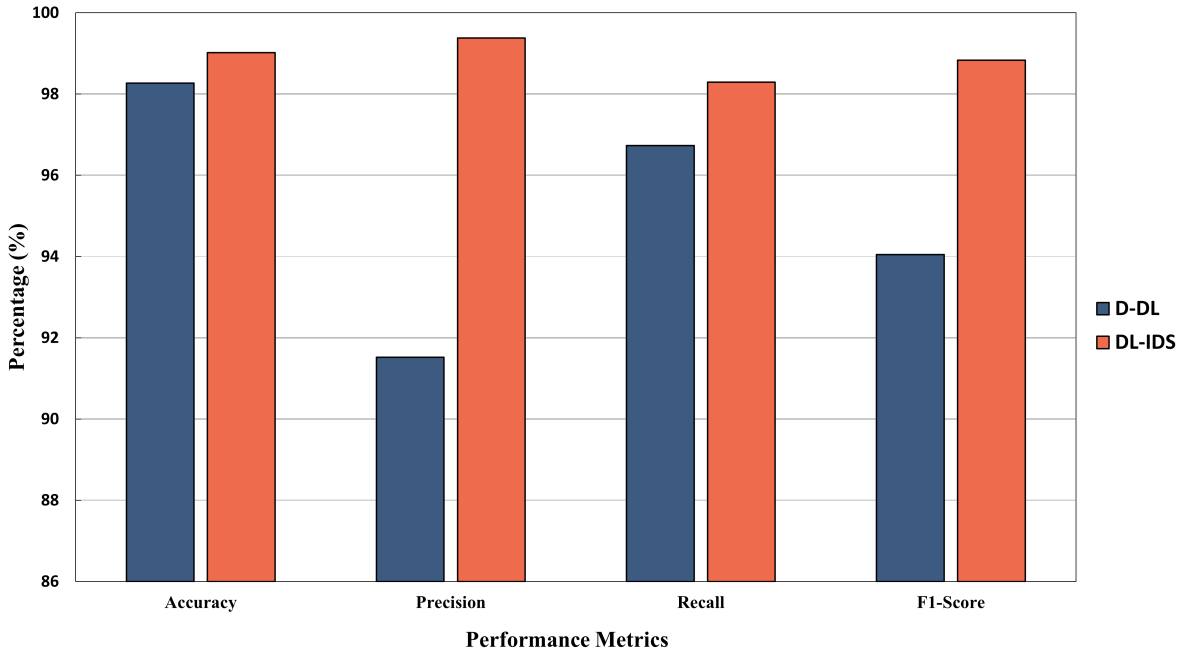

Figure 3.7: D-DL With DL-IDS Performance Comparison

figure also highlights the trade-off between precision and recall in the D-DL, where high recall and low precision mean most positive instances were correctly recognized (low FN). However, IDS identified activity results in many false positives (i.e. false alarms), when in reality they are acceptable behavior.

For any classifier, the F1-score is critical to measure its performance since it gives a harmonic mean of precision and recall metrics for the model. In Figures 3.6 and 3.7, we compared the DL-IDS F1-score with other models and achieved better F1-scores. Therefore, our proposed DL-IDS with nearest neighbour-based preprocessing, SMO-based feature selection and SDPN-based classification provides better security for IoT environments than previous models. In Table 3.4, we compare the performance metrics values of our proposed DL-IDS with existing works (DFEL) and Distributed DL(D-DL).

Table 3.4: Performance comparison on NSL-KDD dataset

Model	***Accuracy (%)***	***Precision (%)***	***Recall (%)***	***F1-Score (%)***
DL-IDS	**99.02**	**99.38**	**98.91**	**99.14**
D-DL	98.27	88.85	96.50	92.52
DFEL GBT	98.54	98.54	98.53	98.53
DFEL KNN	98.82	98.82	98.82	98.82
DFEL DT	98.77	98.77	98.77	98.77
DFEL SVM	98.86	98.86	98.87	98.86

3.4 Summary

Due to the growing number of IoT based networks, this chapter proposed a novel, deep-learning based intrusion detection system (IDS) to identify severe anomalies. With DL-IDS, a spider monkey optimization (SMO) algorithm is used to extract the most relevant features from the dataset. A stacked deep polynomial network (SDPN) is then applied to identify the optimal features and classify the data as normal or anomalous in different attack categories (e.g. DoS, U2R, R2L and probe). We evaluated our DL-IDS system using the NSL-KDD dataset, and it achieved superior results in accuracy (99.02%), precision (99.38%), recall (98.91%) and F1-score (99.14%).

Chapter 4

Anomaly and Signature-based IDS for IoT

4.1 Introduction

The IoT is a massively extensive environment that can manage many diverse applications. Security is critical due to potential malicious threats and the diversity of the connectivity. Devices can protect themselves and detect threats with the IDS. IDS typically uses one of the two approaches: anomaly-based or signature-based. This chapter proposes a model (known as "AS-IDS") [97] that combines these two approaches to detect known and unknown attacks in IoT networks. The proposed model has three phases: traffic filtering, preprocessing, and the hybrid IDS. In the first phase, the arrival traffic is filtered at the IoT gateway by matching packet features, after which the preprocessing phase applies a Target Encoder, Z-score, and Discrete Hessian Eigenmap (DHE) to encode, normalize and eliminate redundancy, respectively. The hybrid IDS integrates signatures and anomalies in the final phase. The signature-based IDS subsystem investigates packets with Lightweight Neural Network (LightNet), which uses Human Mental Search (HMS)

for traffic clustering in the hidden layer. Boyer Moore is used to searching for a particular signature in the output layer that is accelerated by using the Generalized Suffix Tree (GST) algorithm, and by matching the signatures, it classifies the attacks as an intruder, normal or unknown. The anomaly-based IDS subsystem employs deep Q-learning to identify unknown attacks, and uses the Signal-to-Noise Ratio (SNR) and the bandwidth to classify the attacks into five classes: DDoS, Probe, U2R, R2L, and normal traffic. Detected packets are then generated with new signatures, using the Position Aware Distribution Signature (PADS) algorithm. The proposed AS-IDS is implemented in real-time traffic with the NSL-KDD dataset, and the results are evaluated in terms of DR, FAR, Specificity, F-measure and computation time.

IoT is used for many applications, including smart cities, industries and medical services. The applications deliver huge volumes of traffic to the end devices through the network, and since they deal with sensitive and non-sensitive data [98, 99], the attacks on the infrastructures are increasing. The detection of such attacks is a challenge since there is minimal prediction efficiency for the various diverse attack methods. To counter this, an IDS was developed to monitor and analyze network traffic and make decisions regarding the network packets [100–102]. In the simplest case, the arriving traffic is analyzed by extracting the features in the packets and differentiating normal from abnormal traffic using specific features, such as source IP address, destination IP address, source port number, destination port number and others. Security defence systems can detect different types of attacks, including DoS, Distributed DoS (DDoS) and spoofing. Each attack causes unique behaviour on the network, and the attack targets are different. In general, however, they utilize network resources and degrade channel characteristics.

The IDS is classified into two main types; signature-based and anomaly-based [103, 104]. The signature-based IDS stores a set of attack signatures, and identifies arrival attacks by validating their signature in the database. Thus, the use of signature-based IDS is efficient for identifying known attacks in the network, since the signatures are only those that were previously stored. The anomaly-based IDS can predict unknown attacks by monitoring and analyzing the traffic according to the packet features, which are capable of differentiating normal traffic from attack traffic. The challenges in these two IDSs follow:

1. Signature-based IDS: This type of IDS is suitable for detecting known attacks with signatures already in the database; the absence of signatures in the database allows all unknown attacks.

2. Anomaly-based IDS: This IDS can detect unknown attacks but still requires an effective operating algorithm to analyze the arrived packets correctly.

Signature-based approaches have an advantage over anomaly-based methods, as they are simple and can operate online in real time. To ensure a more security-aware environment, a hybrid IDS was designed that integrates signatures and anomalies [105–107]. The hybrid IDS incorporates machine learning algorithms such as a C5 decision tree, a SVM, an OC-SVM, the k-Nearest Neighbor (k-NN), and others [108–110]. Deep learning algorithms are also used to help detect attacks [111], including CNN and RNN. These deep learning algorithms improve detection efficiency. To test an IDS system that uses machine learning algorithms, a network dataset is collected and used to evaluate the system [112, 113]. This dataset is comprised of many packet features from network environments with network devices. The main challenges for IDS systems are:

- Improving the detection rate with accurate attack detection from the dataset using

an efficient algorithm. The increase in the detection rate is critical since it requires a precise analysis of the packet features.

- The signature-based IDS approach can only detect known attacks (i.e. stored), making it challenging to detect unknown ones.

- Slow processing of an algorithm to detect attacks degrades performance since the IoT has huge volumes of packets. The detection of abnormal packets needs to be faster and more efficient.

Challenges in IDSs are addressed in this work, and a hybrid IDS (AS-IDS) has been proposed that can operate quickly and efficiently to detect attacks. This also means the proposed AS-IDS also reduces the detection time.

4.2 The Proposed Model

The proposed AS-IDS model combines signature-based and anomaly-based IDSs, which can detect both known and unknown attacks. The model is comprised of three phases; traffic filtering, preprocessing and the hybrid IDS phase.

Figure 4.1 illustrates the proposed AS-IDS model and shows the proposed used algorithms. IoT gateway has the capability to perform the filtration for the arriving traffic by verifying the main traffic parameters and filtering out the mismatched packets. This process is applied to real-time traffic by matching the dataset features, and the benchmark dataset enters to perform preprocessing and training in the hybrid IDS phase. In the preprocessing phase, the dataset features are decoded and normalized, and redundancies are removed. After preprocessing, the dataset enters the hybrid IDS phase that integrates signatures and anomalies. In hybrid IDS, the signature is matched in LightNet using an HMS algorithm from the constructed signatures tree. The signature-based IDS

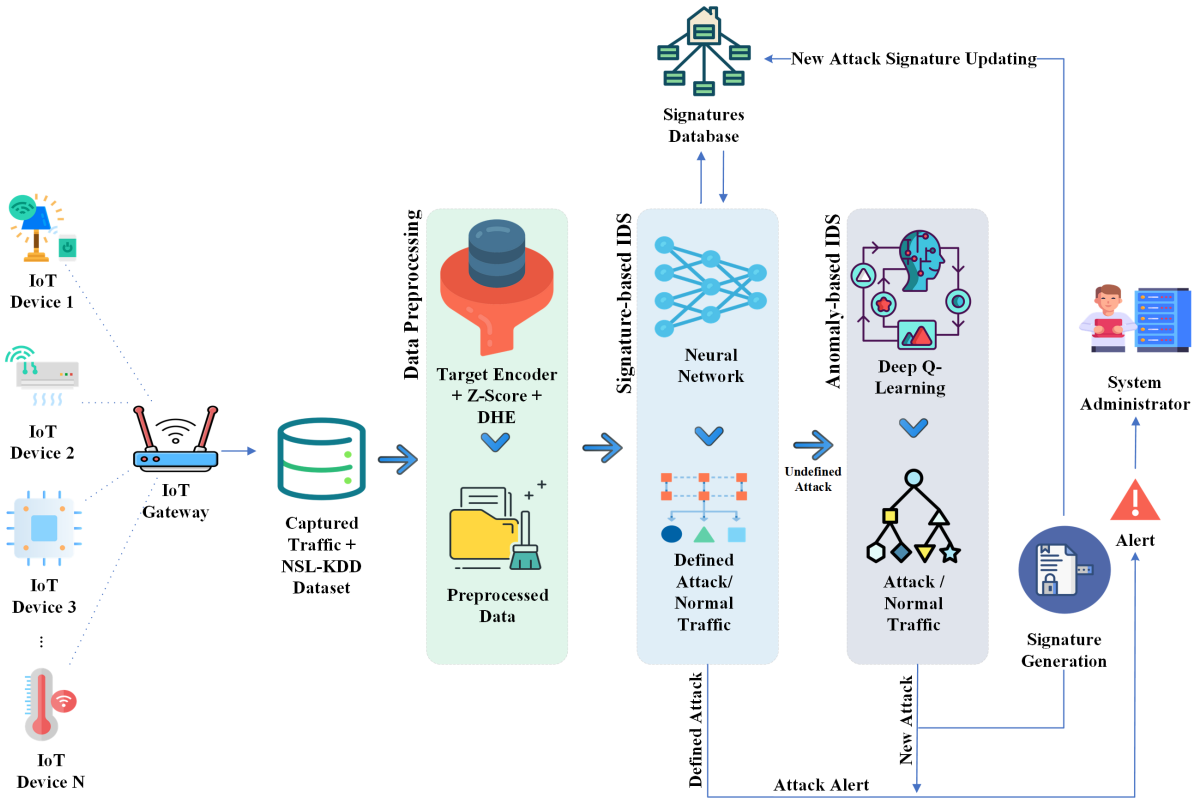

Figure 4.1: AS-IDS Model

analyses identify all known attacks, and unknown attacks are identified by an anomaly-based IDS using a Deep Q-learning algorithm that can learn from the environment. Due to reinforcement learning, if a new packet arrives, it can still be predicted by the IDS. Hence, the proposed AS-IDS is efficient, and deep learning ensures asymptotically optimal performance when high traffic volumes enter the system.

4.2.1 Traffic filtering and Preprocessing

The first phase in the proposed system is traffic filtration of the arriving traffic. The filtration process is handled by validating the source IP address, destination IP address, port numbers, protocol and packet count. Using these features, the arrived traffic is matched in the gateway, where abnormal features are filtered out, and basic attacks

are blocked. For each unfiltered packet passed through the IoT gateway, the SNR and bandwidth are computed by the gateway. Assuming an IoT device is IoT_i and it submits the packet P_i= { S_{IP},D_{IP},P_{no},P_t,P_{ct} } which represent the source IP address, destination IP address, port number, protocol and packet count, respectively. These packet features are cross-verified by the gateway, then move on to preprocessing. The packet filtering pseudo-code is as follows:

Algorithm 3: Packet Filtration

```
Begin;
For all (IoT_1,...,IoT_i);
if P_i (S_IP,D_IP,P_no,P_t,P_ct )= Predefined Values then
    Allow Packet P_i from IoT_i
else
    Discard Packet
end if
End
```

In the second phase, the preprocessing of the dataset starts by encoding the string values in the dataset into numeric values using the Target Encoder algorithm; for example, the protocol type field has TCP, UPD or ICMP string values. The Target Encoder algorithm groups the data by category, track the number of occurrences of each target and calculates the probability of each Target based on the computed mean value. The target fields in the dataset are converted into numeric values. After conversion to numeric values, a Z-score is used to normalize the dataset. The following mathematical formula calculates the Z-score:

$$Z = (X - \mu)/\ \sigma \tag{4.1}$$

Here X, μ and σ represent the original feature vector, observed mean and standard

deviation values, respectively. Using this simple normalization, the dataset fields reach the range of [0-1], improving the classification process. To remove redundancies in the dataset, which can increase processing time and degrade classification performance results, we adapted the DHE [114], which uses H-functions. Thus, the Hessian matrix, coordinate function and constant function identify the similarities between values, reduce the dimensionality by determining the local neighbours and compute the tangent coordinates. This way, the dataset dimensionality is reduced in the preprocessing phase, and attacks are detected by analyzing the packet features.

4.2.2 Signature-based and Anomaly-based IDS

The hybrid IDS subsystem has two main processing sections that combine signature-based and anomaly-based IDS. The signature-based IDS is performed first to detect all known attacks by matching the stored signatures. The signatures are generated from the Position Aware Distribution Signature (PADS) algorithm [115]. The signature is maintained in the repository using a GST, which can match signatures in an asymptotically optimal time.

Let L and M be two signatures the suffix tree is built for. The new signature is generated as $L\#M\$$, where # and \$ represent the suffixes of L and M. If the size of the signatures is m, it uses $O(m+n)$ to match the signature repository. The LightNet algorithm [116] is used to detect known attacks in the signature-based IDS sub-model. HMS [117] is applied to cluster at the hidden layer, and the Boyer Moore algorithm searches the output layer. The LightNet is designed from continuous weight networks. Most of the weight values are 0, and non-zero weights are limited to two, either -1 or +1. This algorithm follows the synaptic pruning training process, and odd or hyperbolic tangent expressions represent the activation function in LightNet. Thus, the arbitrary

location is defined as:

$$\tanh\left(x-\rho\right)\ +\tanh\left(-\left(x-\rho\right)+\chi\right) \tag{4.2}$$

where, $\rho, \chi \in \mathbb{R}$. LightNet is comprised of three layers: input, hidden and output. Packet features are considered to be input, and the hidden layer HMS is used to cluster similar packet features. This HMS algorithm proposes two main processes: searching by Levy flight or by grouping. The Levy flight process is performed based on the following Levy Distribution expression:

$$L\left(x\right)=\frac{1}{\pi}\int_{0}^{\infty}\exp\left(-\alpha q^{\beta}\right)\ \cos\left(qx\right)dx \tag{4.3}$$

Here, α denotes the scaling factor, β is the distribution index that is limited to $0<\beta\leq 2$. Then, the generation of step size is given as:

$$S=\left(2-itr*\left(\frac{2}{itr}\right)\right)*\alpha\bigoplus L(x) \tag{4.4}$$

where itr represents the number of iterations, and the product $\oplus$ means entry-wise multiplications. The similar signatures of the dataset are clustered with the clustering K-means algorithm, then the k-value is determined, and the signatures are clustered in the hidden layer. After this, the output layer is responsible for searching using the Boyer Moore pattern matching algorithm, which is an effective suffix heuristic process since it conducts bad character and good suffix approaches. The bad character is the character of the text which does not match the current character of the pattern. Based on this mismatch, the pattern is shifted until the mismatch becomes a match or the pattern passes the mismatched character.

With the good suffix approach, the signature string matches the pattern by the fol-

lowing four steps:

Step 1: Signature S in pattern P matches at time t;

Step 2: Pattern P with the prefix matches the suffix;

Step 3: The P moves all the characters to S; and,

Step 4: It generates match or mismatch results in S.

The hybrid IDS phase applies a signature-based IDS that detects known attacks by matching signatures in the tree. Using LightNet, the received packets are classified into three classes: intruder, normal and unknown attack. The intruder packets are reported, and the unknown packets are analyzed in the anomaly-based IDS to identify the attack types precisely.

The classification of the signature-based IDS is then carried out by an anomaly-based IDS, and only the unknown attacks are processed. In anomaly-based IDS, a deep Q-learning algorithm is used that considers SNR and bandwidth parameters and classifies the attack as DoS, Probe, U2R or R2L. With Q-learning, the agent learns the environment and generates a Q-table matrix with the states (Features) and actions (Send/Not Send Alert). However, Q-learning is only suitable for small-scale environments, and the IoT is an exceedingly large-scale environment. Thus, Q-learning is combined with deep learning to become a Deep Q-learning algorithm that can simultaneously process multiple unknown attack packets.

Each packet consists of SNR and bandwidth as input to the input layer of the Deep Q-learning algorithm. The benefit of using reinforcement learning is that it evaluates the previous result and can determine optimal future actions. In contrast, deep learning is the optimal classifier algorithm suitable for large input volumes. The combination of these allows Deep Q-learning to apply the classification.

Let the states and actions be represented as $(S_1, S_2, S_3, \ldots . S_t)$ and $(A_1, A_2, A_3, \ldots . A_t)$ respectively. The Q-value in this Deep Q-learning is determined from the following expression as:

$$\mathrm{Q}(S_t, A_t) \leftarrow Q\left(S_t, A_t\right) + \alpha\left[R_{t+1} + \gamma \max_{\alpha} Q\left(S_{t+1}, a\right) - Q(S_t, A_t)\right]$$

where R_{t+1} is defined as a reward by 1 on each timestep, according to the attack detection decision. Thus, the learning agent in this reinforcement learning algorithm learns policy $\pi(A_t|S_t)$. If γ is the learning rate, S_t and A_t are the state and action for that specific packet, respectively. An epsilon-greedy policy is applied in deep Q-learning to perform the actions. According to the proposed algorithm, the states depend on the SNR and bandwidth of the packet and other significant packet features to detect four attacks (i.e. DoS, Probe, U2R, R2L) that are unspecified in the signature-based IDS.

Table 4.1: State-Action Pairs

States	**Actions**	**New (State,Action)**
$\boldsymbol{S_1}$	A_1	S'_1, A'_1
$\boldsymbol{S_2}$	A_2	S'_2, A'_2
$\boldsymbol{S_3}$	A_3	S'_3, A'_3
$\boldsymbol{S_4}$	A_4	S'_4, A'_4
$\vdots$	$\vdots$	$\vdots$
$\boldsymbol{S_t}$	A_t	S'_t, A'_t

Table 4.1 details the states and actions that are defined and, due to agents' ability to learn, new states and actions are defined that make future predictions more accurate. The nodes in Deep Q-learning know which previous decision-making experience is key to improving future decisions. Thus, the loss function is predicted from the mean square error Q-value and the target Q-value. Attacks are differentiated and detected based on deviations of the major elements of the packet features.

Algorithm 4: Anomaly-based IDS

```
1 Begin;
2 For all (S_t,A_t);
3 if S' is terminal then
4 |   Compute new initial state from the reward
5 else
6 |   S←S';
7 |   Return attack type {DoS,Probe,U2R,R2L}
8 end if
9 End
```

For detected attacks, the signature is generated and updated in the repository tree, as it is essential to eliminate attacks in the signature-based IDS when it occurs in the future. This is done using signature generation PADS, which contain segments of both anomalous and standard signatures.

The byte frequency distribution of the traffic is computed and compared with the distribution of normal traffic [115]. A large difference is considered anomalous. Anomalous signature positions the signature length w with respect to the byte frequency distribution, where W is the width of the signature in terms of the number of bytes.

After the signature is generated, it is updated in the repository, maintained as a tree. From the anomaly-based IDS, it is classified as Dos, Probe, U2R, R2L or normal. Thus, the proposed AS-IDS model ensures efficient attack prediction from the packets and network parameters, which are also crucial for intruder detection. A network with numerous possible intruders will decrease network performance in terms of limited resource utilization, longer transmission times, wasted channels and other factors. Thus, detecting intruders and attackers by the network dataset will help increase network performance.

4.3 Experimental Evaluation

In this section, the experimental evaluation of the proposed model is compared with previous models. This includes simulation setup, dataset description, comparative analysis and highlights. The NSL-KDD dataset (described in detail in Section 3.3.1) is used to evaluate the performance metrics. The efficiency of the proposed model is determined based on the comparisons.

4.3.1 Simulation Setup

The AS-IDS is developed by a network simulator and an IDS dataset that can determine intruder behaviour in the system. The NS3.26-based network simulation is performed on packets from the IoT nodes that are designed to behave as real nodes in an actual network environment. Table 4.2 shows implementation parameters for the simulated network. NS3 for IDS incorporates the proposed algorithms into the system to detect intruders. All the algorithms are written in C++ and called by a Python script. The results are evaluated based on the C++ algorithms as graphic plots of the significant performance metrics.

4.3.2 Comparative Analysis

This section highlights the metrics of the significant constraints of the proposed model and compares them with DBN, and Deep Recurrent Neural Network (DRNN) algorithms that were used previously in the IDSs [53, 118, 119]. The main performance metrics are the detection rate, false alarm rate, sensitivity, specificity and F-measure.

Table 4.2: Implementation Parameters

Parameter	**Range**
Simulation Specifications	
Network Area	250×250m
Number of IoT nodes	50
Number of IoT gateway	1
IDS	1
Mobility Model	Randomwaypoint
Signature based IDS classes	Normal, Intruder, Unknown
Anomaly based IDS classes	Normal, DoS, Probe, U2R, R2L
Number of Packets	100
Packet Interval	1 s
Packet Size	512
Simulation Time	300 s
System Specifications	
Simulator Version	NS3.26
Operating system	Ubuntu-14.04 LTS
System type	32-bit
Processor speed	2.5 GHz
System Processor	Intel Core i7

4.3.3 Detection Rate and False Alarm Rate

DR is defined as the ratio between the numbers of correctly detected events, e.g. attack, and the total number of these events. as shown in the following equation:

$$DR = \frac{TP}{TP + FN} \tag{4.5}$$

where, TP and FN are True Positive and False Negative, respectively. The FAR is defined as the calculated ratio between the number of negative events, e.g. attacks that are incorrectly detected as positive (false positives) and the total number of the actual negative events and expressed as the following equation:

$$FAR = \frac{FP}{FP + TN} \tag{4.6}$$

This parameter should be as high as possible in order for better prediction, and the false alarm rate should be as low as possible in the detection systems to make attack detection efficient. Figures 4.2 and 4.3 show the detection and false alarm rates of Deep-RNN, DBN algorithms and the proposed AS-IDS.

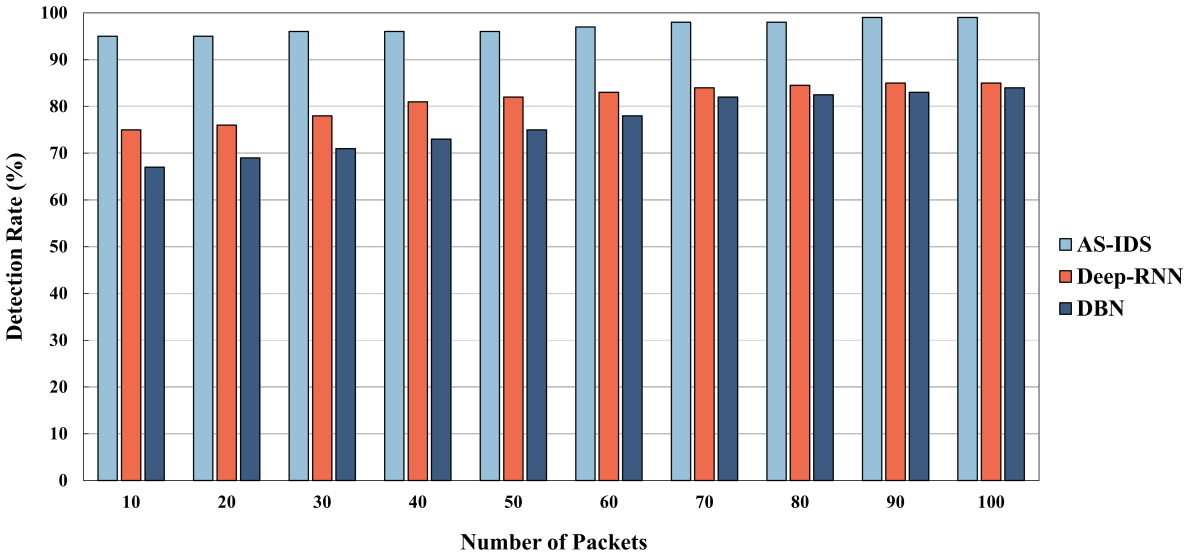

Figure 4.2: Detection Rate Comparison

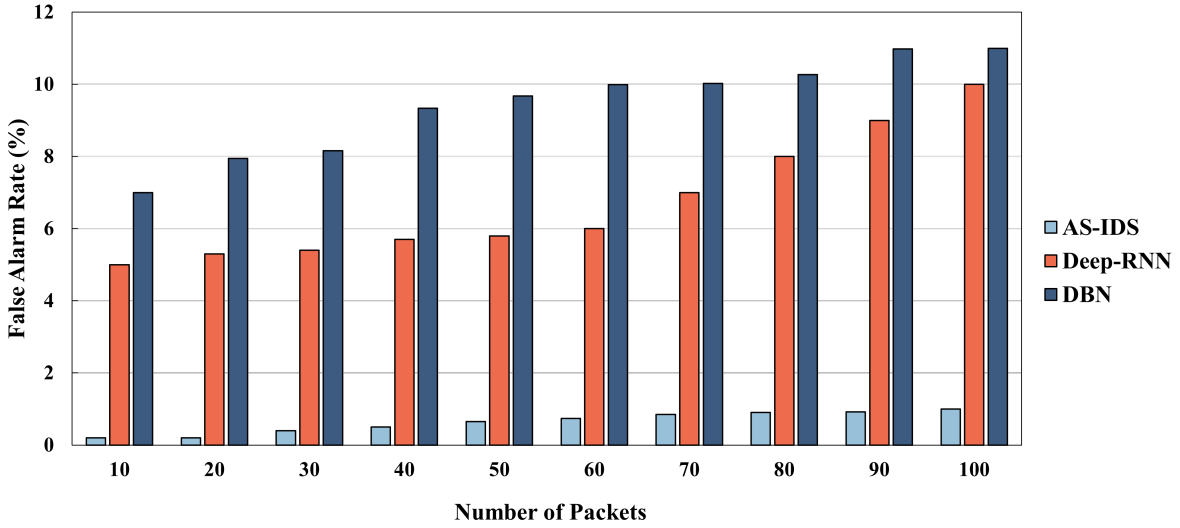

Figure 4.3: False Alarm Rate Comparison

The proposed AS-IDS has a higher detection rate due to the use of signature-based and anomaly-based detection to identify attacks. The matching of signatures is also more accurate due to the use of the Boyer Moore method and anomaly detection by deep learning, as well as the use of environmental parameters that help improve the detection rate.

On average, the detection rate of AS-IDS is 96.9%, which ensures support for the continuous increase of packets in the network. The DBN and DRNN algorithms have lower detection rates of 76.45% and 81.35%, respectively. This is due to the lack of preprocessing, which results in less consideration of the significant features used for the classification and emphasizes the importance of the significant features in the IDS systems, even with deep learning algorithms. The improvement in detection rates will reflect the decrease in false alarm rates in the proposed model. The false alarm rate is defined as the number of attacks that the IDS does not correctly detect. They are reported to the administrator as attacks on the network.

A higher false alarm rate indicates that the model performs poorly in identifying attacks. The reasons for a decrease in detection rate and an increase in false alarm rate are follows:

- The dataset of traffic is collected and used raw, which can introduce redundancy and degrade classification results due to the need to correlate the normal and redundant data. In addition, processing redundant data requires more time since the dataset is larger. Processing IDS using a dataset will always create redundant data, and processing with the redundant data will degrade the system's performance significantly.

- Other models may fail to take significant features into account. This can cause attack packet behaviour to be detected from the features less efficiently since each

attack packet has different features. It is essential to consider the most significant packet features for processing to increase the detection rate.

- Although deep learning algorithms can learn the features dynamically through the training data process, they cannot learn the current environment parameters without using reinforcement learning.

Considering the supporting data, the proposed model performs better than existing works. Therefore, the proposed AS-IDS performs better than the Deep-RNN and DBN algorithms when used with the IDSs.

4.3.4 Sensitivity, Specificity and F-measure

Sensitivity and specificity parameters play a vital role in evaluating IDS performance that classifies attacks. The sensitivity defines a True Positive Rate, and the specificity defines a True Negative Rate. The sensitivity is computed based on the proportion of positive classes made up of attackers and non-attackers. In turn, specificity is computed from the proportion of detected negative attacks from the dataset.

Sensitivity and Specificity performance with respect to packet increases are evaluated, and the results are shown in Figures 4.4 and 4.5. From the investigations of AS-IDS, Deep-RNN and DBN, the proposed AS-IDS shows improvement regardless of the number of arrival packets. The two parameters are computed by mathematical expressions based on the classification results. The sensitivity and specificity are defined as follows:

$$Sensitivity = \frac{N(TP)}{N\,(TP) + N(FN)} \tag{4.7}$$

$$Specificity = \frac{N(TN)}{N\,(TN) + N(FP)} \tag{4.8}$$

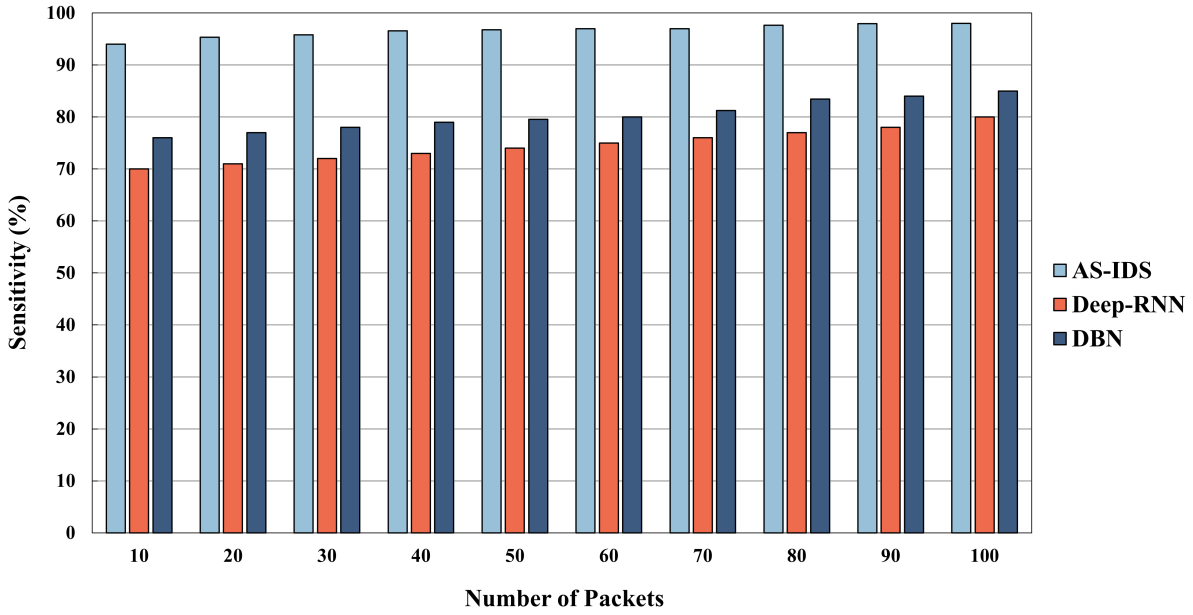

Figure 4.4: Sensitivity Comparison

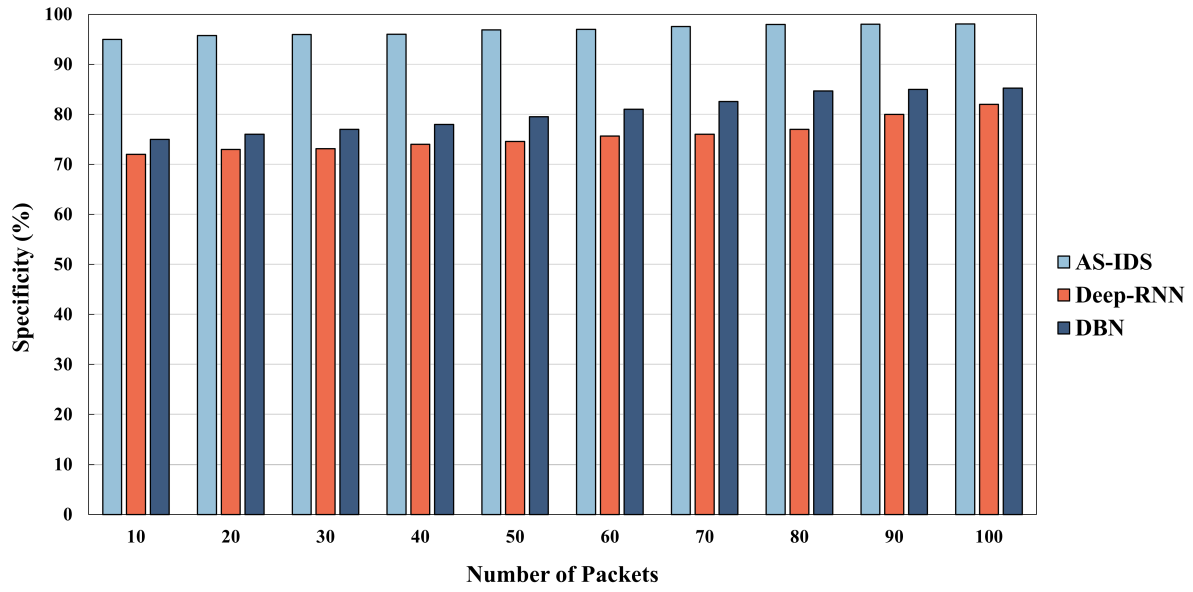

Figure 4.5: Specificity Comparison

where N(TP), N(TN), N(FP) and N(FN) denote the number of true positives, true negatives, false positives and false negatives, respectively. The higher sensitivity and specificity indicate that the proposed system performs better than existing works. Based

on this, the sensitivity represents the precision of the prediction of normal packets, and the specificity identifies the correctness in classifying the attack packets. The overall performance in terms of sensitivity is 96.60% for the proposed AS-IDS, and 74.60% and 80.32% in the algorithms, Deep-RNN and DBN for IDS, respectively. The difference between existing works is 22% and 16.2%, which means the proposed AS-IDS functions are superior to other deep learning methods. This is due to the preference tendency for significant features and improved layer processing in LightNet. In addition, preprocessing results in improved sensitivity. Similarly, the specificity results show growth with respect to increasing numbers of packets. This increase in specificity is 96.82% for the proposed model and 75.73% and 80.40% in previous methods Deep-RNN and DBN, respectively. This evaluation indicates that the proposed AS-IDS results have higher detection performance than other IDSs. Accuracy of classification results is achieved by estimating the F-measure parameter, which is determined from the classification's true positive, true negative, false positive and false negative values. Figure 4.6 illustrates the performance of the proposed AS-IDS compared to the existing deep learning-based IDS.

4.4 Summary

Reducing high false alarm and false positives rates remains challenging for intrusion detection in IoT environments. None of the works in the literature have focused on managing high stream packets in IDS perception layers, and most work in hybrid and signature-based-intrusion detection models is based on pattern matching algorithms. However, these methods can only work under single packet verification, not HTTP traffic-based environments (IoT). A significant issue with anomaly-based IDS is the inefficient linking of abnormal and intrusive factors. None of the works have concentrated on abnormal inducing factors such as SNR and bandwidth. In this chapter, we proposed a model

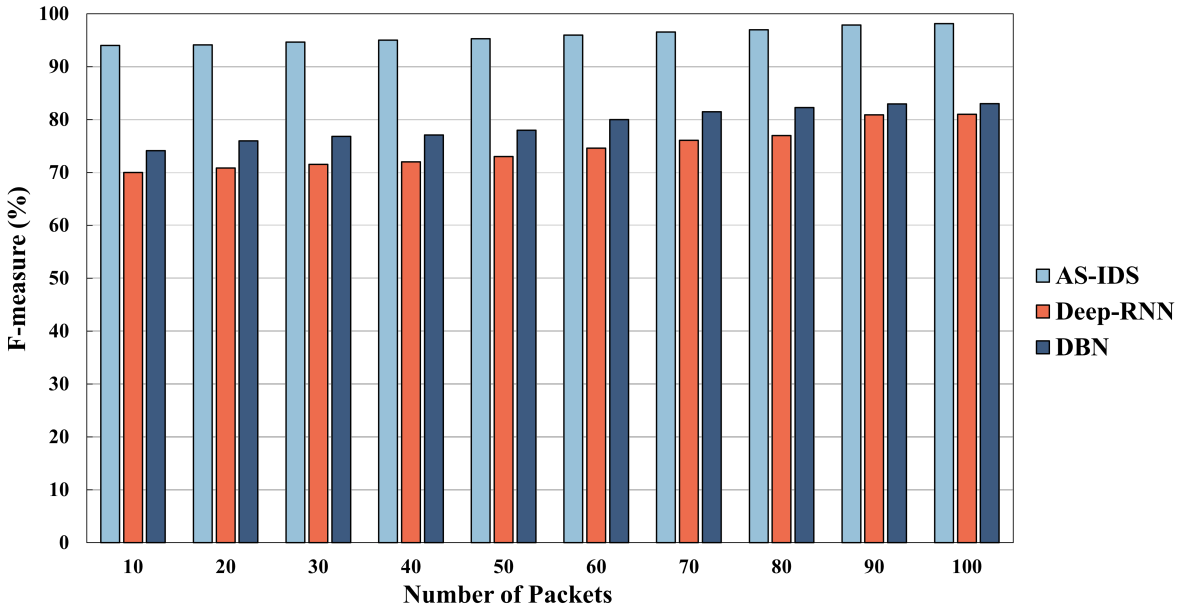

Figure 4.6: F1-Score Comparison

that combines signature and anomaly-based IDS. The three phases considered here are traffic filtering, preprocessing and hybrid IDS. In the traffic filtering phase, the features of the arrived packet streams are extracted and validated by the IoT gateway. In preprocessing, the features are converted into numeric values, normalized, and the redundancy is reduced. Preprocessing concentrates on the network traffic and data previously collected from the dataset. The traffic packets then enter the hybrid IDS phase, where the signature-based IDS is applied using signature matching and the LightNet algorithm. The anomaly-based IDS processes all unknown packets, and the deep Q-learning algorithm considers the SNR and bandwidth for attack classification.

Chapter 5

Transfer Learning-based Intrusion Detection

5.1 Introduction

Transfer Learning (TL) is a method of reusing the acquired knowledge (i.e. parameters) from previously trained models and applying it to another model. This technique can be helpful when the available dataset used for the source domain is small compared to the datasets in the target domain, which can also lower the required training time. Diverse applications of transfer learning have been demonstrated in IoT fields. In Section 5.2 of this chapter, we use transfer learning to transfer the knowledge between two different benchmark datasets; CICIDS2017 and CSE-CIC-IDS2018, in which TL is applied to two different deep learning algorithms, namely DNN and CNN. In addition to achieving satisfying performance and reduced training time for the target domains, our analysis illustrates the computational effectiveness of the proposed model by transferring the knowledge from the smaller to the larger datasets.

In Section 5.3, we applied the TL technique to design an IDS and used the IoV networks as a use case. The IoV is the trending technology in ITS that offers safe and efficient driving, enhances traffic flow, and gives a cleaner environment. IoV has two types of communications: intra-vehicle communication (e.g., connecting Electronic Control Unit (ECU) devices with the other subsystems) [120] and inter-vehicle communication (e.g., Vehicle-to-Vehicle (V2V), Vehicle-to-Pedestrian (V2P), Vehicle-to-Infrastructure (V2I), Vehicle-to-Cloud (V2C) and Vehicle-to-satellite (V2S) communication) [84],[121]. In IoV, the vehicle has different internal components; the two main components are the Controller Area Network (CAN) and the ECU. The CAN is a bus topology central network inside the vehicles that connects different ECUs and other I/O devices. The ECU is the central computation unit in the vehicle which controls other connected systems and subsystems inside the vehicle, such as Braking Systems, Anti-lock Braking System (ABS), Airbags, Tire Pressure Monitoring systems (TPMS), and cruise control [122]. Our proposed model has been shown to outperform other standard ML/DL models used in the same environment, such as RNN, DNN, SVM, and Adaptive Boosting (AdaBoost) and achieves a detection accuracy of 93.95%, and 94.51% in NSL-KDD and CICIDS2017 datasets, respectively.

5.2 Transfer Learning Technique Between Two Different Datasets

This section proposes a framework that uses multi-task transfer learning to transfer knowledge gained from two different benchmark datasets (CICIDS2017 and CSE-CIC-IDS2018). To the best of our knowledge, this is the first work that uses transfer learning to transfer the knowledge between two different datasets. The performance of the intru-

sion detection engine is evaluated using two different deep learning algorithms, namely Deep Neural Network (DNN) and Convolutional Neural Network (CNN). In addition to achieving satisfying performance and reduced training time for the target domains, our analysis illustrates the computational effectiveness of the proposed model by transferring the knowledge from the smaller to the larger dataset.

5.2.1 CICIDS2017 Dataset

CICIDS2017 dataset [123] contains real Packet Captures (PCAP's) and is considered as one of the most up-to-date IDS datasets to contain normal traffic and some common attacks, such as brute force attacks, Heartbledd attacks, botnet, DoS attack, DDoS attack, web attack and infiltration attacks. The attacks are distributed with normal traffic for different days (Monday, Tuesday, Wednesday, Thursday, and Friday), whereas the Monday dataset has only normal traffic. In addition, the CICIDS2017 dataset contains labelled network traffic, such as timestamp, source, and destination IPs, source and destination ports, and protocols generated by the network traffic flow generator and CICFlowMeter analyzer. The attacks on Wednesday with the DDoS attacks on Friday are used to evaluate the performance of our proposed model. The number of normal and attack instances on the Wednesday dataset and DDoS attack from Friday are shown in Tables 5.1 and 5.2, respectively.

5.2.2 CSE-CIC-IDS2018 Dataset

CSE-CIC-IDS2018 dataset [124] contains about 16 million instances collected through different ten days. It has 80 network traffic features and seven main types of attacks: DDoS, DoS, Botnet, Brute-force, Infiltration, Web attack, and Botnet attack. These represent more than 60% of the total number of instances. Table 5.3 shows the num-

Table 5.1: Attacks in CICIDS2017 dataset

Days	**Normal and Attacks Labels**
Monday	Benign
Tuesday	BForce, SFTP and SSH
Wednesday	DoS Hearbleed, DoS Slowloris, DoS Slowhttptest, DoS Hulk and DoS GoldenEye
Thursday	Web BForce, Web XSS, Web Sql Injection, Infiltration Dropbox Download and Cool disk
Friday	DDoS LOIT, Botnet ARES, and PortScans (sS, sT, sF, sX, sN, sP, sV, sU, sO, sA, sW, sR, sL and B)

Table 5.2: Number of Normal and Attacks Instances in CICIDS2017-Wednesday Dataset

Attack Label	**No. of Instances**
Benign Traffic	440031
DoS Hulk	231073
DDoS	41835
DoS GoldenEye	10293
DoS slowloris	5796
DoS Slowhttptest	5499
Heartbleed	11

ber of normal traffic and attacks instances in both CSE-CIC-IDS2018 and CICIDS2017 datasets.

5.2.3 Preprocessing

In this scenario, we use the two benchmark datasets: CICIDS2017 and CSE-CIC-IDS2018. The prepossessing for both datasets was performed by mapping the attack labels in both CICIDS2017 and CSE-CIC-IDS2018 datasets. Table 5.4 shows the corresponding attacks in each dataset—and mapping the features of the two datasets. CICIDS2017 and

Table 5.3: Normal traffic and attacks distributions in CSE-CIC-IDS2018 and CICIDS2017 datasets

Instance	**CICIDS2017**	**CSE-CIC-IDS2018**
Benign Traffic	1,743,179	6,112,151
DDoS Attack	128,027	687,742
DoS Attacks	252,661	654,301
Botnet Attacks	1966	286,191
Brute-force Attacks	13,835	380,949
Infiltration Attacks	36	161,934
Web Attacks	2180	928
Port Scan Attacks	158,930	-

CSE-CIC-IDS2018 datasets have 83 and 79 features, respectively, whereas the four extra features (Flow ID, Source IP, Source Port, and Destination IP) in the CICIDS2017 can be dropped since they are least important.

5.2.4 Model Construction

The DNN and CNN algorithms were used to build the source models using the CICIDS2017 dataset as the source domain. By analyzing the two datasets, we uncovered that using the CICIDS2017 dataset as the source domain needs less training time thanks to fewer samples and gives better performance, as discussed later in Section 5.2.7 of this chapter. A feature representation that maps the original feature space into a shared subspace between two or more datasets can be used to transfer knowledge using Neural Networks. By doing so, insufficient data instances will be dealt with, and training time will be reduced—consequently, The more similar the domains for which the TL can be used, the better the model will perform. The DNN network has six layers: one input layer, four hidden layers of 32, 16, 8, and 4 nodes, and a Softmax layer which serves as a classification layer. On the other hand, the CNN has eight layers: one input, one

Table 5.4: CICIDS2017 and CSE-CIC-IDS2018

CSE-CIC-IDS2018 Sub-dataset	**Corresponding CICIDS2017 Sub-dataset**	**CSE-CIC-IDS2018 Attacks**	**Corresponding CICIDS2017 Attacks**
CSE-CIC-IDS2018-Feb 14	CICIDS2017-Tuesday	FTP-BruteForce	FTP-Patator
		SSH-Bruteforce	SSH-Patator
CSE-CIC-IDS2018-Feb 15,16	CICIDS2017-Wednesday	DoS GoldenEye	DoS-GoldenEye
		DoS slowloris	DoS slowloris
		DoS Slowhttptest	DoS-SlowHTTPTest
		DoS Hulk	DoS Hulk
CSE-CIC-IDS2018-Feb 22,23,28 + CSE-CIC-IDS2018-Mar 1	CICIDS2017-Thursday	Web Attack - Brute-force	Brute-force -Web
		Web Attack – XSS	Brute-force -XSS
		Web Attack – Sql Injection	SQL Injection
		Infiltration	Infiltration
CSE-CIC-IDS2018-Feb 20,21 + CSE-CIC-IDS2018-Mar 2	CICIDS2017-Friday	Botnet	Botnet
		DDoS LOIT	DDoS attacks-LOIC-HTTP
		DDoS-LOIC-UDP	N/A
		DDOS-HOIC	N/A
N/A	CICIDS2017-Friday	N/A	Port Scan
N/A	CICIDS2017-Wednesday	N/A	Heartbleed Attack

Convolutional1D, one Max pooling1D, one Flattens layer, one Dropout, two dense layers, and an output layer. Using the DNN and CNN algorithms, the source model is trained on the CICIDS2017 dataset and then assigned to the local models on the edges. With edge datasets, the local models are then trained using cross-entropy loss based on the following equation:

$$L(\Theta) = -\sum_{i=1}^{k} y_i \log(\hat{y}_i) \tag{5.1}$$

where the number of input training samples is i, and the value of the label is y. After training the local datasets with the source models, the new aggregated model will be composed without disclosing the user's personal data, based on the parameters of the newly trained models; a similar approach is applied in [125]. In order to calculate an aggregated model, the local models are averaged, and it is represented as follows:

$$M_g(x) = \frac{1}{N} \sum_{k=1}^{N} M_{lk}(x) \tag{5.2}$$

where N is the number of local models, and x is the parameter set. Edge datasets can directly be assigned to the aggregated model. However, it will not perform well because it learns only the shared features across all datasets and not those customized for each dataset. In this case, a more efficient way to do transfer learning is to freeze all layers, replace the output layer with a new output layer, and then unfreeze the layers individually according to the performance desired. During the backpropagation process, the model freezes the first few layers that hold the low-level features from each edge dataset and tries to learn the more specific features from the other layers. Adding a new edge dataset will allow a more efficient generalization of the source model. Furthermore, the source model can be updated through synchronization with the edge models and passing the updated models to all edge nodes to apply transfer learning and create a new personalized local model.

5.2.5 Layers Freezing Technique

One of the techniques used to enhance the model performance and decrease the training time in the transfer learning is the layer freezing, in which we freeze the gained parameters of some of the top layers of the pre-trained model that have the generic features and only update the other layers' parameters that have the specific high-level features (fine-tune).

Our experiments show the effectiveness of freezing the layers for two different cases: freezing the first layer and freezing the first two layers. More details of the experiment for the layers freezing are shown in Section 5.2.7.

5.2.6 Knowledge Transfer

In this step, the knowledge gained from the previous step will be transferred and reused to build a new local model with the CSE-CIC-IDS2018 sub-datasets in the same way as the training algorithms mentioned in the previous step. The Inductive Multi-Task Transfer learning approach has been used since the source, and the target domains (features) are the same. Multi-task learning is a particular type of multi-output learning [126] that learns different related tasks with different training datasets or features at the same time with the aim to enhance the generalization performance or increase the relationship between those tasks. However, multi-output learning uses the same training set. The newly trained local models are transferred back to be aggregated to build a new source model using the transferred parameters from the local models. The used datasets share the same feature spaces but differ in the instances of source and target task labels (attacks). As a result, each local edge uses a customized model constructed by combining the aggregated source model with the local data. So, the best possible performance is achieved while maintaining data privacy (closest to that obtained by directly training the model).

5.2.7 Performance Analysis

To evaluate the performance of the detection engine, we consider the performance metrics accuracy, precision, recall, F1-score, and the average of training time and fine-tuning time (The time needed to tune the model for the new task). The dataset CICIDS2017 was

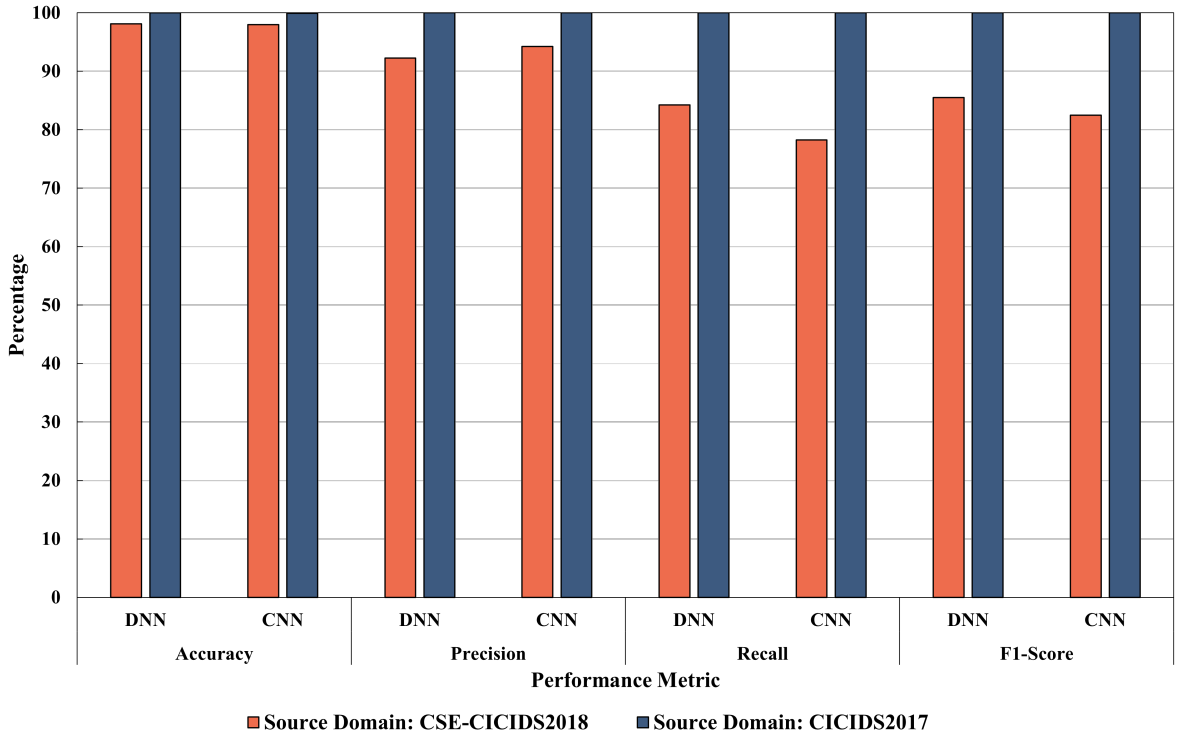

Figure 5.1: Model performance for all attacks when freezing first layer

chosen to train the source model using the DNN and CNN algorithms for two main reasons. First, this dataset has fewer than 3 million instances compared to the CSE-CICIDS2018, which has more than 16 million instances, which also illustrates the model's effectiveness in transferring the gained knowledge from the relatively small dataset to the large dataset. Second, as shown Figure 5.1 and Figure 5.2, we found by the experiments that using CICIDS2017 to train the source model gives a better performance for the following two cases. The first case is shown in Figure 5.1 where the first layer is frozen, and the other layers' parameters were tuned. The second case shown in Figure 5.2 shows the effects of freezing the top two hidden layers and tuning the parameters in the other hidden layers. We took the average performance for the selected attacks (DoS, DDoS, Botnet, and Brute-force) in both cases.

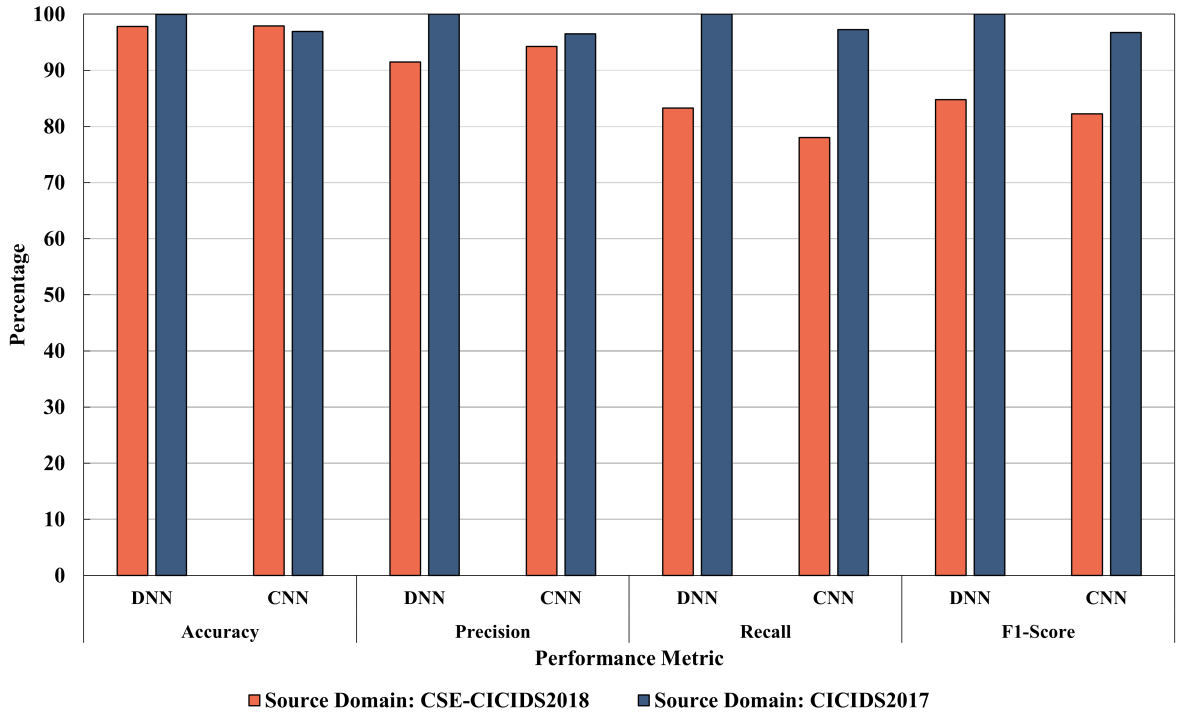

Figure 5.2: Model performance for all attacks when freezing top two layers

In Figure 5.4, we show the average performance of freezing the top three layers of the DNN and CNN algorithms when using both datasets. The performance contrast between the two algorithms when freezing different layers results from the fact that DNN and CNN algorithms have 9617 and 48951 trainable parameters, respectively. Those parameters will be decreased when freezing some of the layers and after fine-tuning the parameters in the remaining layers. Based on the experiments, we conclude that DNN works better using 177 trainable parameters when freezing the top two layers by achieving 97.83% and 99.97% accuracy when using CSE-CICIDS2018 and CICIDS2017 datasets as the source model, respectively. On the other hand, CNN works better using 279 trainable parameters that are satisfied with freezing the first layer by achieving 97.96% and 99.90% accuracy when using CSE-CICIDS2018 and CICIDS2017 datasets as the

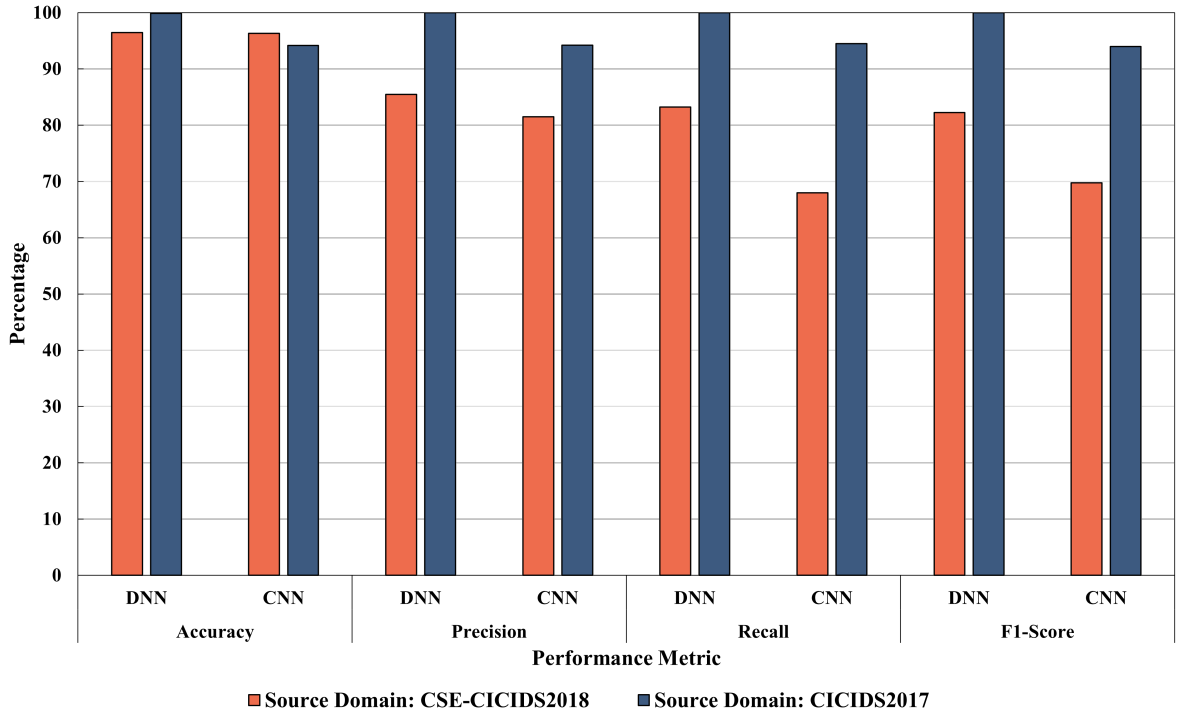

Figure 5.3: Model performance for all attacks when freezing top three layers

source model, respectively. However, the minimum required trainable parameters for satisfactory performance will be 41 and 39 for DNN and CNN, respectively. To support this argument, we calculate the average training and fine-tuning times for both algorithms in Table 5.5, which shows how the time needed to customize the model is significantly less than training it directly. Performance and training time must be balanced to maximize the model's performance; a better model performance needs more trainable parameters and takes more training time. However, those times could also be affected by machine capabilities. We implemented the algorithms using Keras/Pytorch running as a Python library on TensorFlow cloud servers and a workstation with an Intel Core i7 processor and 16GB DDR4 RAM.

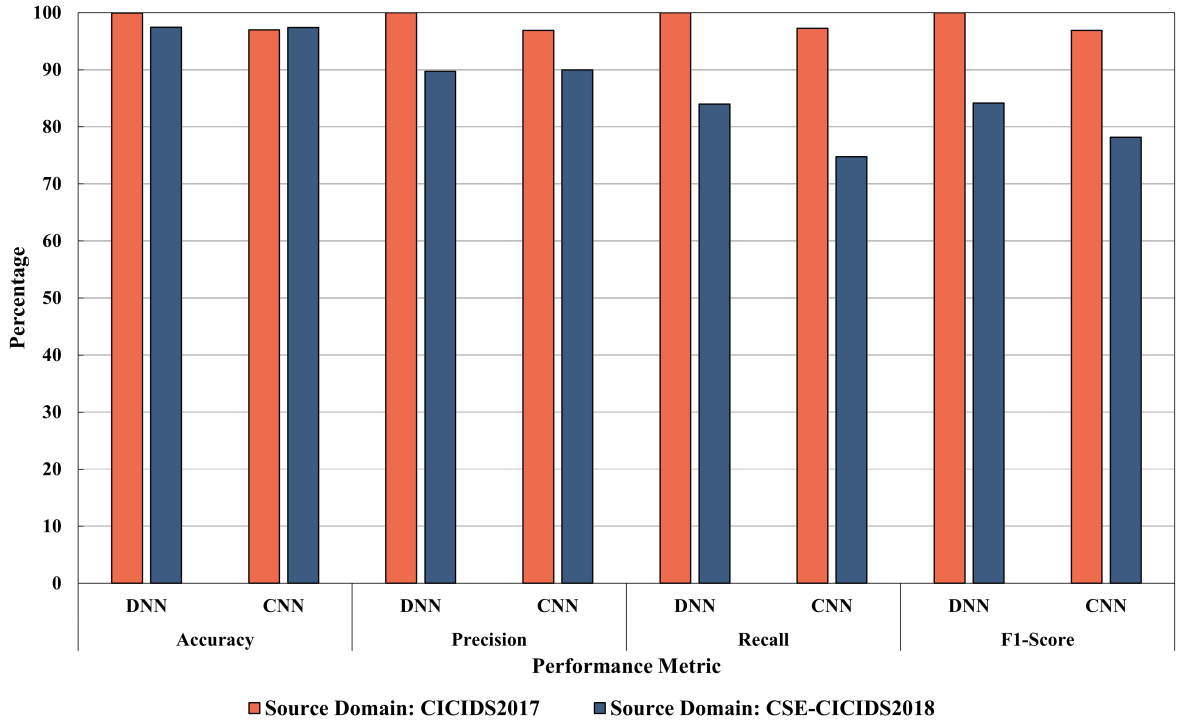

Figure 5.4: Model average performance for all attacks

5.3 Transfer Learning-based IDS (Use case: Internet of Vehicles (IoV))

5.3.1 Model Workflow

Due to different forms of communications in the IoV, Figure 5.5 shows our view of an IoV infrastructure; this technology is subject to different types of cyberattacks or threats, such as DDoS and Distributed Denial of Service (DoS), spoofing and sniffing attacks, etc., in which the attackers can disrupt the vehicle operation, switch off the engine or malfunctioning the brake system, causing serious issues with the vehicles or passengers, and affect road safety. One of the significant attacks that an IoV is vulnerable to is the

Table 5.5: Training and Fine Tuning Time Comparison

	Algorithms			
	DNN		**CNN**	
Attacks	**Avg. Training Time (S)**	**Avg. Fine Tuning Time (S)**	**Avg. Training Time (S)**	**Avg. Fine Tuning Time (S)**
DoS	732.56	595.09	826.41	647.83
DDoS	916.65	894.79	854.35	789.90
Botnet	368.05	164.21	313.44	236.83
Brute-force	223.71	114.29	312.81	239.60

DoS and DDoS attack in which the connected vehicles can be used as a botnet to attempt these types of attacks and cause road congestion and increase trip duration [127]. One of the ways that can be used to secure the IoV environment is the IDS. However, it has some challenges when it comes to designing and developing it for the IoV. For example,

- Choosing the proper placement for the security device or software in the IoV infrastructure (e.g., host-based or network-based). The centralized approach offers the advantage of centralized management but can also lead to system processing overload, which may affect the Quality of Service (QoS) in a connected vehicular network. The distributed strategy can reduce the amount of monitored traffic and increase the processing capacity. However, implementing an IDS in different regions of a connected vehicular network is challenging due to the associated management issues.

- The second challenge is the limited resources for the vehicle components, such as memory, data transmission rate (bandwidth), and computational resources.

- The real-time limitation must be considered when proposing the IDS for any security solution.

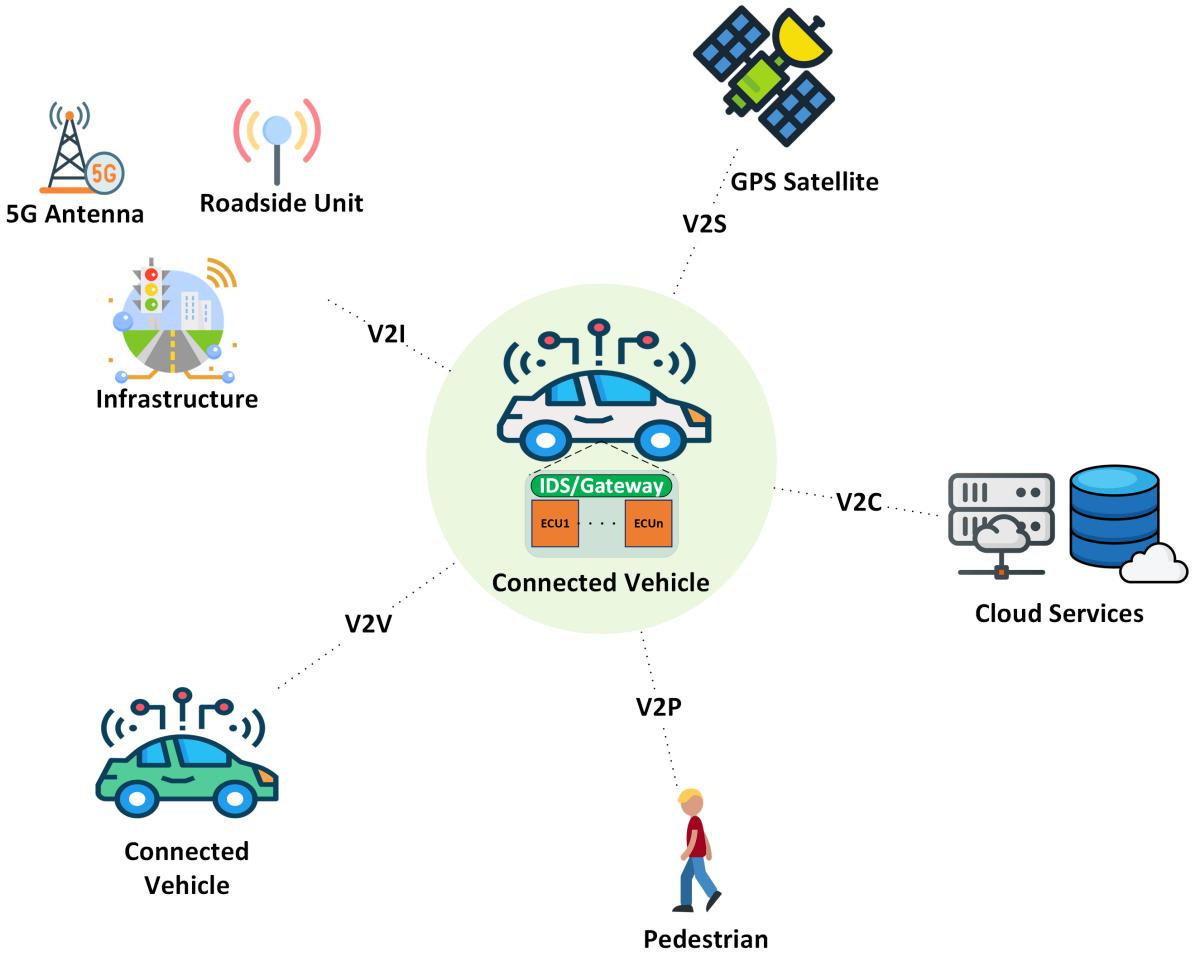

Figure 5.5: Internet of Vehicles Infrastructure

Therefore, to address some of the challenges mentioned above of protecting the IoVs, we propose an IDS using DBN and RF algorithms to detect the attacks in both intra- and inter-vehicle communications. The main contributions of this model are the following:

- Support the anomaly-based IDS with a signature-based IDS subsystem that helps to work in the real-time IoV environment by storing the previously detected attack signatures on the vehicles that are generated from the cloud security management system.

- Adapt using the over-the-air update concept to update the signatures of the attacks

from a cloud-based security management system so each vehicle connected to the network will work as a packet inspector that helps to find the attack signatures and add them to the cloud signature database, which needs less memory storage and more computation resources.

- Adapt using the Self-taught Transfer Learning (STL) to transfer the knowledge of a pre-trained DBN model from the source domain, which saves training time and increases the accuracy of the detection engine.

This proposed model places the IDS in the connected vehicles. Figure 5.6 shows the workflow for the proposed model. A signature-based detection first screens the traffic by matching the stored patterns in the signature table, and in case some pattern is matched, an alert will be triggered to the driver and the security management system administrator. In case no attack is detected, the traffic will be classified by the detection engine using the DBN classifier. In the event of a new attack, the log of this attack will be sent to a cloud-based security management system that basically contains the attack signature generator and a database with complete information about the attack signatures and each connected vehicle's signature list. The system will send an Over-The-Air-Update (OTAU) with the new attack signature to all the network-connected vehicles for new attack signatures. The general OTAU concept refers to distributing new software updates, configuration settings, or encryption keys from a central unit to all the connected devices or users, and those devices or users can accept or reject these updates. The automobile companies have used the OTAU for the connected vehicles to distribute the critical software and security updates to the vehicles after they are sold [128]. To the best of our knowledge, signature-over-the-air-update is used for the first time in this model. In this case, each vehicle connected to the network will work as a packet inspector that helps find the attack signatures and contribute to the cloud signature database and

then generalize it to the newly connected vehicle framework to reduce the false positive attack detection rate.

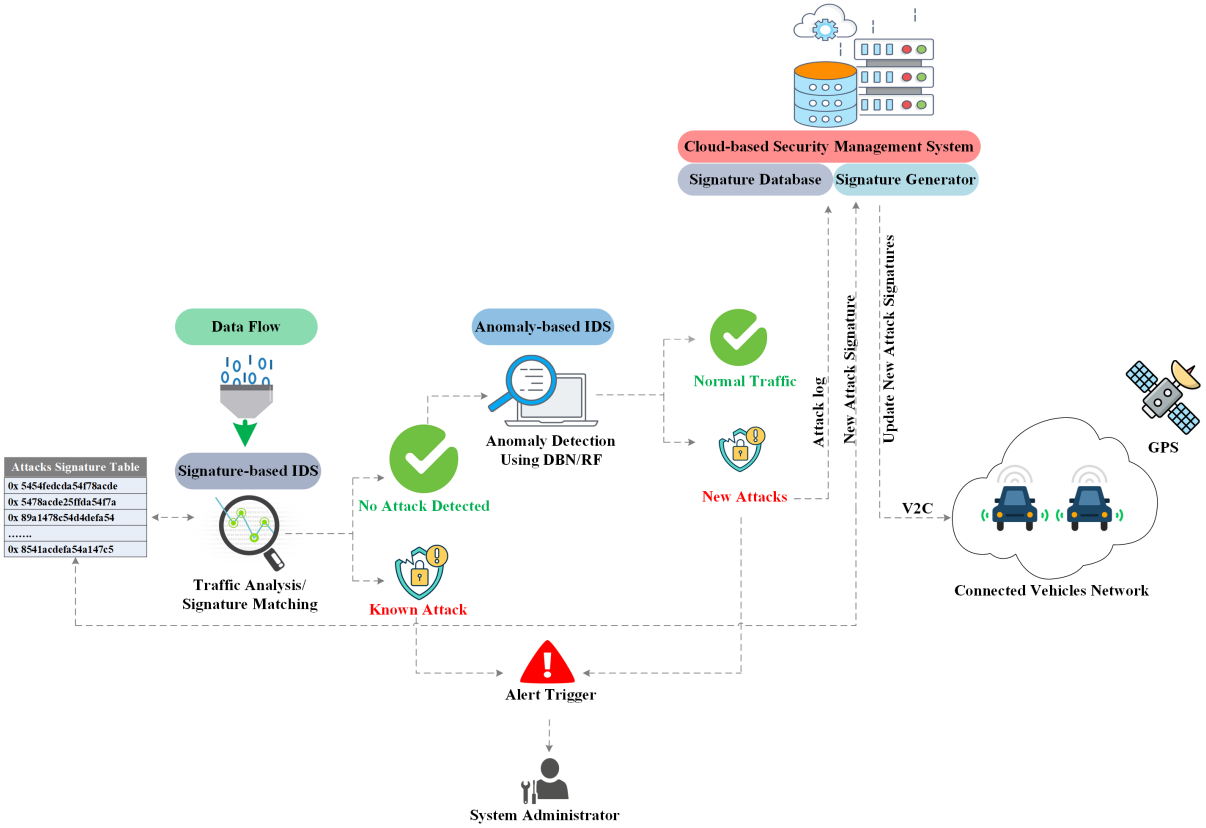

Figure 5.6: Proposed Model Flowchart

5.3.2 Random Forest Algorithm for Feature Selection

In this model, we use the two well-known datasets, NSL-KDD (described in Section 3.3.1) and CICIDS2017 (described in Section 5.2.1). This model uses the RF algorithm for feature selection. RF is an embedded feature selection method that combines the filtering and wrapping methods used for feature selection. The feature selection helps reduce the model over-fitting, boost the speed of the training process by reducing the model complexity, and increase the model performance. RF uses random tree-based schemes that rank the significant features based on their capability to enhance the Gini node

impurity measure. Given a node t and estimated class probabilities $p(k|t)$, $k = 1, \ldots, N$, the Gini importance index $G(t)$ is defined as:

$$G(t) = 1 - \sum_{k=1}^{N} p^2(k|t) \tag{5.3}$$

where the N is the number of classes. The sum of Gini indexes over all nodes in a tree represents the feature importance value in that tree. The overall importance value for a feature in the forest is defined as the average of its importance value through all trees in the forest. Table 5.6 shows the main tuning parameters with their values used in the RF to select the dataset features. The importance of each feature related to the attacks selected by the RF for the CICIDS2017 and NSL-KDD datasets are shown in Figures 5.7 and 5.8, respectively. We settled for these features in this chapter due to the space limitation, but the reader can refer to [129] and [95] for a complete list of features.

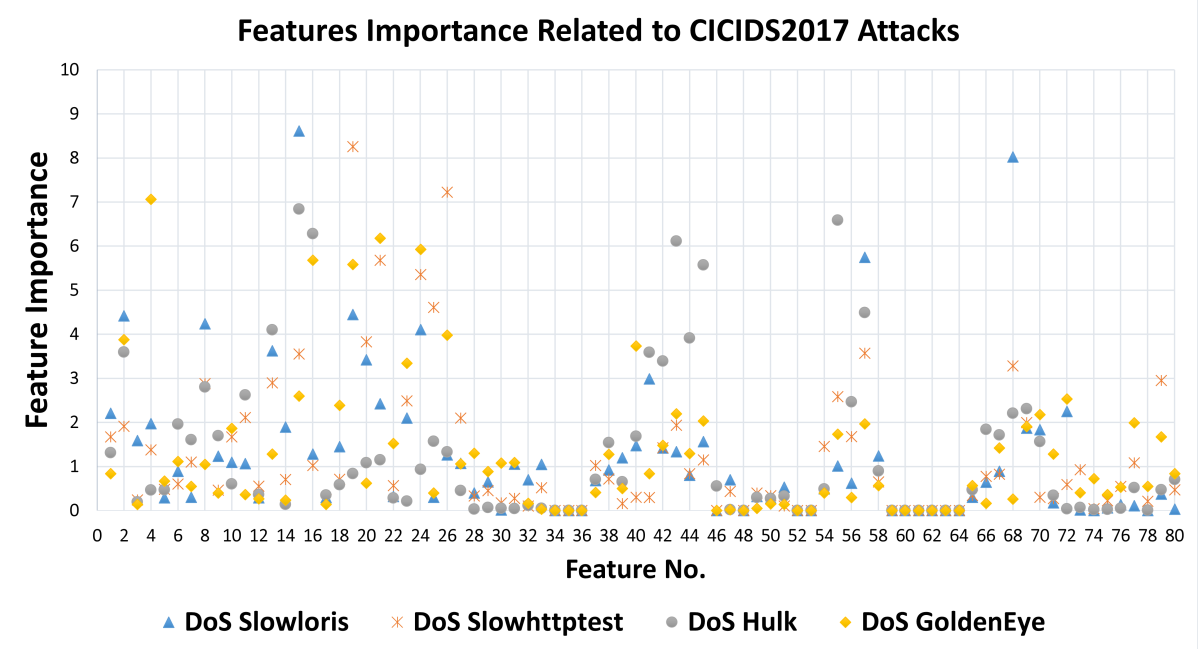

Figure 5.7: CICIDS2017 features importance related to the attacks

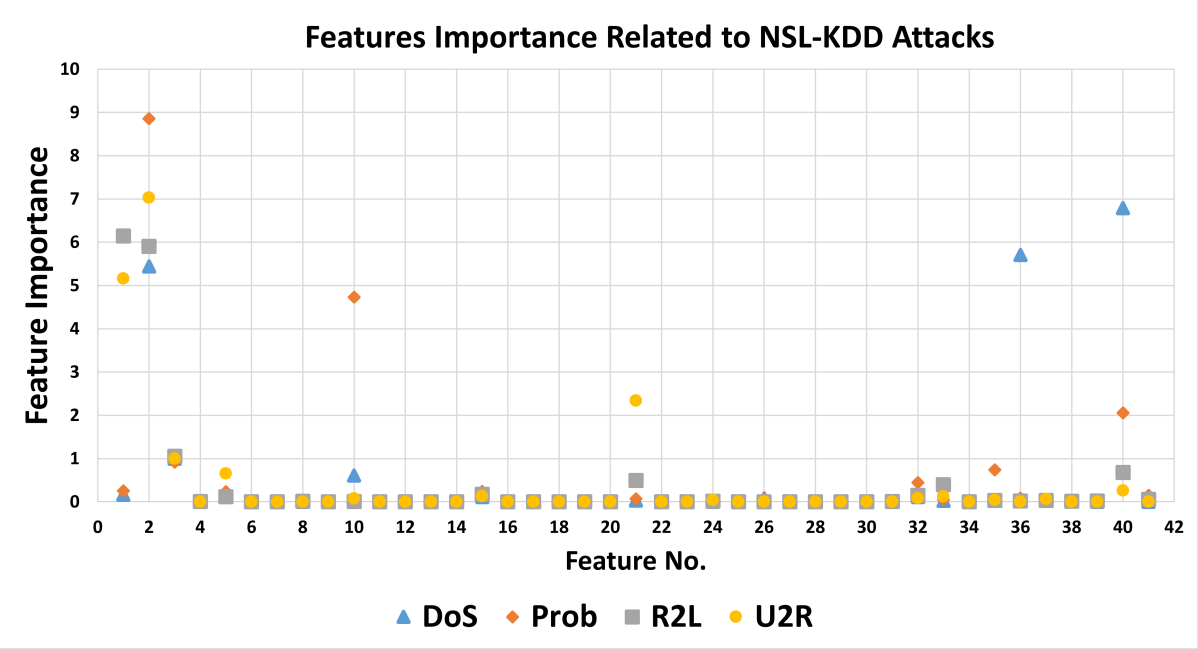

Figure 5.8: NSL-KDD features importance related to the attacks

Table 5.6: Random Forest Parameters

Parameter	Value
No. of estimators	200
Criterion	Gini
Maximum depth	None
Min_samples_split	2
Min_samples_leaf	1
Max_features	122, 80
Warm_start	False
Verbosity	True
Max_leaf_nodes	None
Bootstrap	True
Maximum No. of Samples	None

5.3.3 Attacks Detection

Selected features in the previous phase are fed as input to the intrusion detection engine. In this phase, we applied the idea of TL to reuse the gained knowledge (parameters

and features) from a source to a target domain. A comprehensive survey of transfer learning can be found in [130]. TL has different approaches based on the technique used to transfer the knowledge, such as Inductive Transfer Learning (ITL), Unsupervised Transfer Learning (UTL), and Transductive Transfer Learning (TTL). In ITL, the source and target domains (feature spaces) in the two models should be similar, but the source and target tasks (label spaces) should differ. However, the ITL is of two types: multitask learning and self-taught learning. In this proposed model, we applied the self-taught learning and reused the pre-trained DBN model to transfer the knowledge from the source domain and construct a feed-forward DNN. The DBN consists of un-directed connections among the layers, wherein every layer consists of non-connected n-neural nodes. At the same time, the NN is a kind of feed-forward NN with multiple layers. An unsupervised learning algorithm can be used to train the DBN layers initially and create the DNN model. Thus, DNN can be developed from a pre-trained model using unsupervised learning, which is faster than supervised learning. The DBN consists of different stacked Restricted Boltzmann Machines (RBM) that contain visible nodes, v, and a layer of binary hidden nodes, h. Each node takes the random value of 0 or 1, based on the following probabilities:

$$p(h_i = 1|v) = f(b_i + W_i v) \tag{5.4}$$

$$P(v_i = 1|h) = f(a_i + W_i h) \tag{5.5}$$

The total energy of this combined design of visible and hidden nodes (v,h) is given by:

$$E(v, h) = -\sum_{i,j} v_i h_i W_{ij} - \sum_i v_i a_i - \sum_j h_j b_j \tag{5.6}$$

where i represents the visible layers, j the hidden layers, and $w_{i,j}$ indicates the weight

connection between the i^{th} visible and j^{th} hidden node. Furthermore, v_i and h_j represent the state of the i^{th} visible and j^{th} hidden node, respectively, and a_i and b_j represent the biases of the visible and hidden layers, respectively. The RBM defines a joint probability over the hidden and visible layers as follows:

$$p(v, h) = \frac{e^{-E(v,h)}}{Z} \tag{5.7}$$

where Z is the partition function, calculated by adding the energy of all combinations of visible and hidden nodes (v, h), $Z = \sum_{v,h} e^{-E(v,h)}$. Furthermore, for every transaction in the network, the label (y) and a binary classification layer will be inserted at the top layer of the DBN in order to create a DNN. The DNN model is now trained with an upstream supervised learning method using the label y. During the supervised learning procedure, every node in a DNN layer is allocated with a weight parameter that is manipulated using the gradient descent technique. The DNN framework employs supervised training and binary classification for identifying the attacks. If the DNN model detects an unknown attack, then the relevant tuple of the filtered features will be stored in the cache as feedback. This feedback will be used during retraining the DNN, thus enhancing labelling functionality and feature extraction of the detection engine. A five-layer DL of the proposed model is shown in Figure 5.9, consisting of one input layer, three hidden layers, and one output layer (binary classifier layer). The input layer representing the network features are fed to the DNN. These input features with label y will be fed to the DNN, thereby passing it through the first hidden layer and filtering out the x significant features during the supervised training. These x features will then be passed into the second hidden layer, filtering out z features. The second layer will then feed them into the third hidden layer. These z features will be taken as input by the third hidden layer filtering two outputs, acting as a softmax layer, tuning the results in order to classify the attacks.

The result will then be passed into the output layer, signifying the classification as benign and malicious traffic. In the output layer, there will be no filtering; instead, the output from the third hidden layer will be ingested, and classification results will be obtained. Thus, every layer of the DNN feeds onto this data, mapping it to a numerical value.

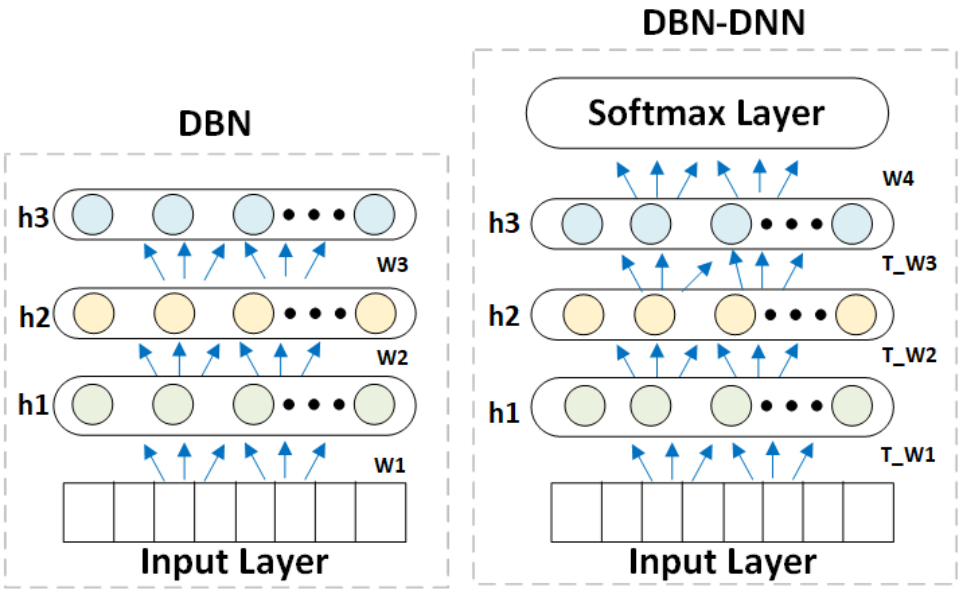

Figure 5.9: Proposed Model Structure

5.3.4 Performance Evaluation

In this model, we considered for comparison the well-known metrics for the classification techniques, such as Accuracy, Precision, DR, F1-score, and the Receiver Operating Characteristic (ROC) curve between the proposed model and four different ML/DL classifiers, namely RNN, DNN, SVM, and Adaptive Boosting (AdaBoost). Eight attacks were used to evaluate the model's performance, namely DoS Hulk, DDoS, DoS GoldenEye, DoS Slowloris, DoS Slowhttptest, Prob, U2R, and R2L from the CICIDS2017 and NSL-KDD datasets. The performance results for the NSL-KDD dataset are shown numerically and graphically in Table 5.7 and Figure 5.10, respectively. They show that the proposed

model outperforms the other baseline ML/DL models shown in the table for the NSL-KDD dataset due to the use of the best 24 out of 41 significant features set chosen by the RF, which improved the detection accuracy from 87.70% to 93.95% and reduced the training time by about 14% without RF. On the other hand, the performance results for the CICIDS2017 dataset are shown in Table 5.8 and Figure 5.11, respectively. The results indicate that the proposed model reaches 94.51% detection accuracy using the most significant 30 features out of 81 selected by RF. The best accuracy without the RF was 90.23%. However, our model reduced the training time by about 11% under the same settings.

Another performance indicator is the ROC curve, which is a graphical representation showing the relationship between the True Positive Rate (aka. sensitivity or recall) and the False Positive Rate at different thresholds. A more effective detection system should have higher and lower false alarm rates. The Figures 5.12 and 5.13 show the ROC curves for the NSL-KDD and CICIDS2017 datasets, respectively. We can see how the proposed model gives a curve closer to the top-left corner of the graph, indicating better performance with a higher detection rate and lower false alarm rate among other models using the two different datasets. Moreover, the two figures show that our proposed model has the best Area Under the Curve (AUC) values compared with other models. It has an AUC of 0.9695 and 0.9568 in NSL-KDD and CICIDS2017, respectively, which indicates that our model has the best ability to distinguish attacks from normal traffic among other models considered.

5.4 Summary

In this chapter, we used the transfer learning technique to transfer the knowledge between two different benchmark datasets: CICIDS2017 and CSE-CIC-IDS2018. By applying two

Table 5.7: Performance metrics on NSL-KDD dataset

Algorithm	**Accuracy%**	**Precision%**	**DR%**	**F1-Score%**
Proposed Model	93.95	93	94	93
RNN	92.52	93	92	92
DNN	80.37	82	82	80
SVM	73.62	77	76	74
AdaBoost	75.49	79	78	75

Table 5.8: Performance metrics on CICIDS2017 dataset

Algorithm	**Accuracy%**	**Precision%**	**DR%**	**F1-Score%**
Proposed Model	94.51	95	95	94
RNN	89.90	89	91	90
DNN	90.30	92	90	90
SVM	88.69	91	89	89
AdaBoost	92.08	93	92	92

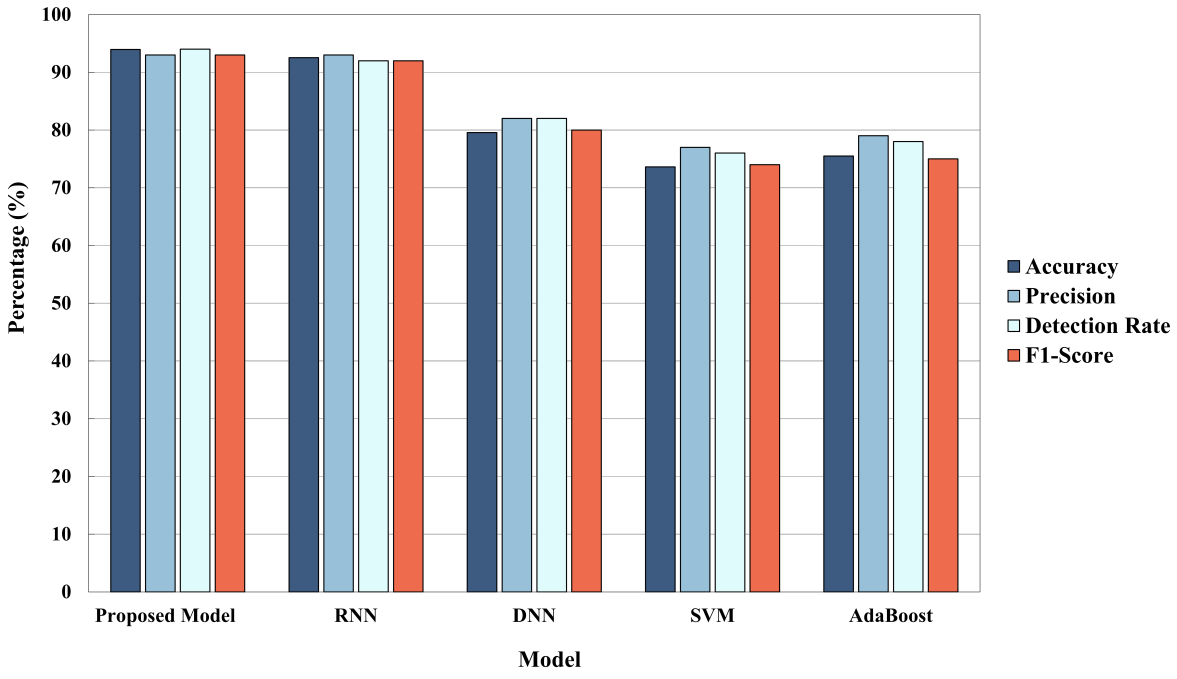

Figure 5.10: Performance Comparison on NSL-KDD Dataset

different deep learning algorithms, namely Deep Neural Network (DNN) and Convolutional Neural Network (CNN), we achieved satisfying performance and reduced training

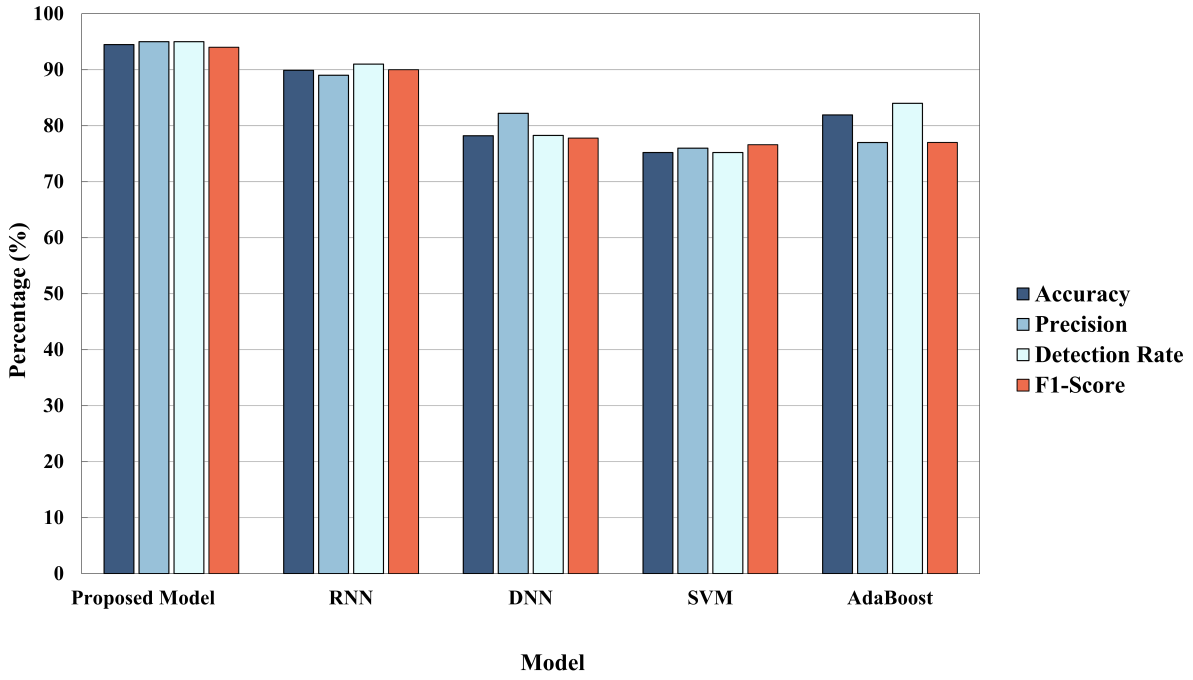

Figure 5.11: Performance Comparison on CICIDS2017 Dataset

time for the target domains. In the second part of this chapter, we utilized the IoV as a use case and proposed an infrastructure-independent IDS to secure the IoV networks. The anomaly-based IDS is augmented with a blacklist of attack signatures placed in the connected vehicles and uses an over-the-air-update concept to update the signatures of the attacks from a cloud database where the signatures of the new attacks can be generated and distributed to the connected vehicles. The detection engine is used the STL to transfer the knowledge of a pre-trained DBN model from the source domain and construct a feed-forward DNN where RF algorithm is used to select the significant features. The CICIDS2017 and NSL-KDD datasets were used to evaluate the performance metrics accuracy, precision, detection rate, F1-score, and ROC curves.

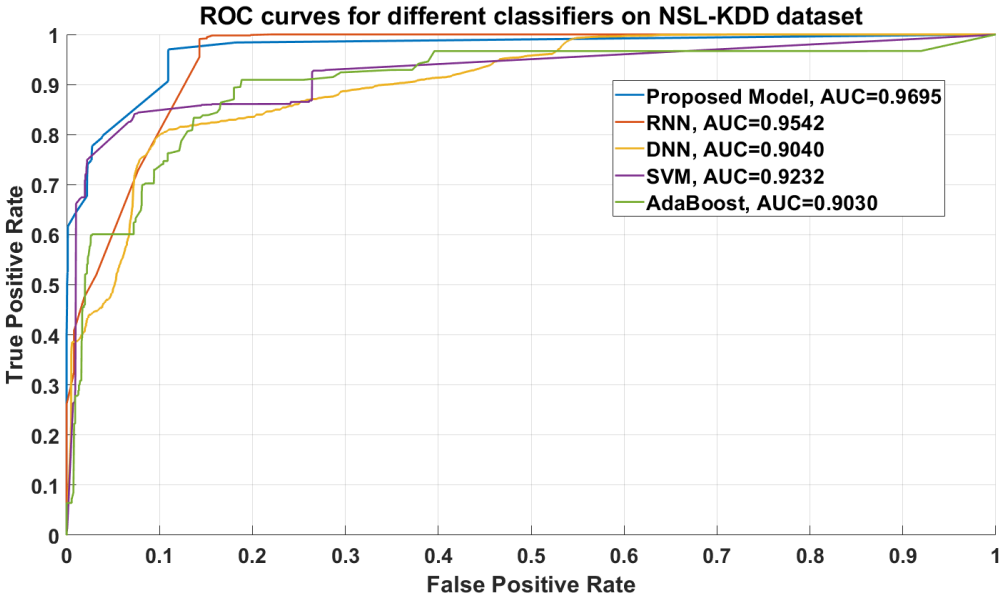

Figure 5.12: Receiver Operating Characteristic Curve (ROC) for Different Classifiers on NSL-KDD Dataset

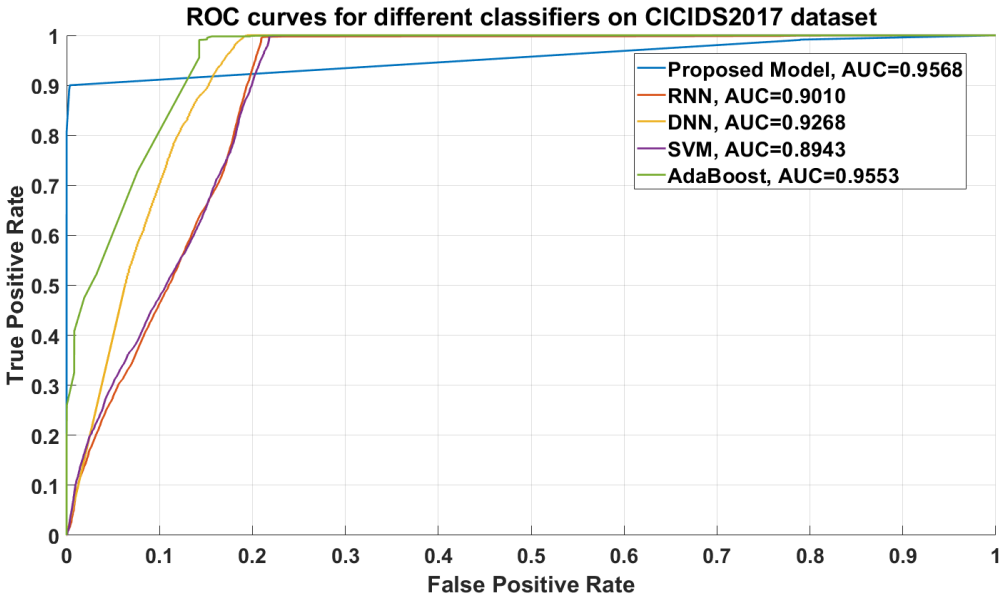

Figure 5.13: Receiver Operating Characteristic Curve (ROC) for Different Classifiers on CICIDS2017 Dataset

Chapter 6

Federated Learning-based Intrusion Detection

6.1 Introduction

FL is an ML technique that supports a global learning platform that collectively utilizes locally available models. FL has recently garnered much attention in many IoT applications because it preserves data privacy [25]. This chapter utilize the idea of FL to design an IDS for the IoMT environments. The IoMT consists of numerous medical devices and applications that are connected to healthcare systems through the Internet. Cyberattacks on those systems are on a steep rise. Online attacks on healthcare organizations, which have been increasing rapidly in recent years, pose a serious and constant threat to day-to-day work and confidential patient data. According to Cybersecurity, Ventures [131], the healthcare industry will fall victim to two to three times more cyberattacks in 2021 than the average numbers for other industries. Cybersecurity Ventures predicts the global healthcare cybersecurity market will grow by 15% every year to reach $125 billion cumulatively over five years from 2020 to 2025. Many hospitals and clinics now

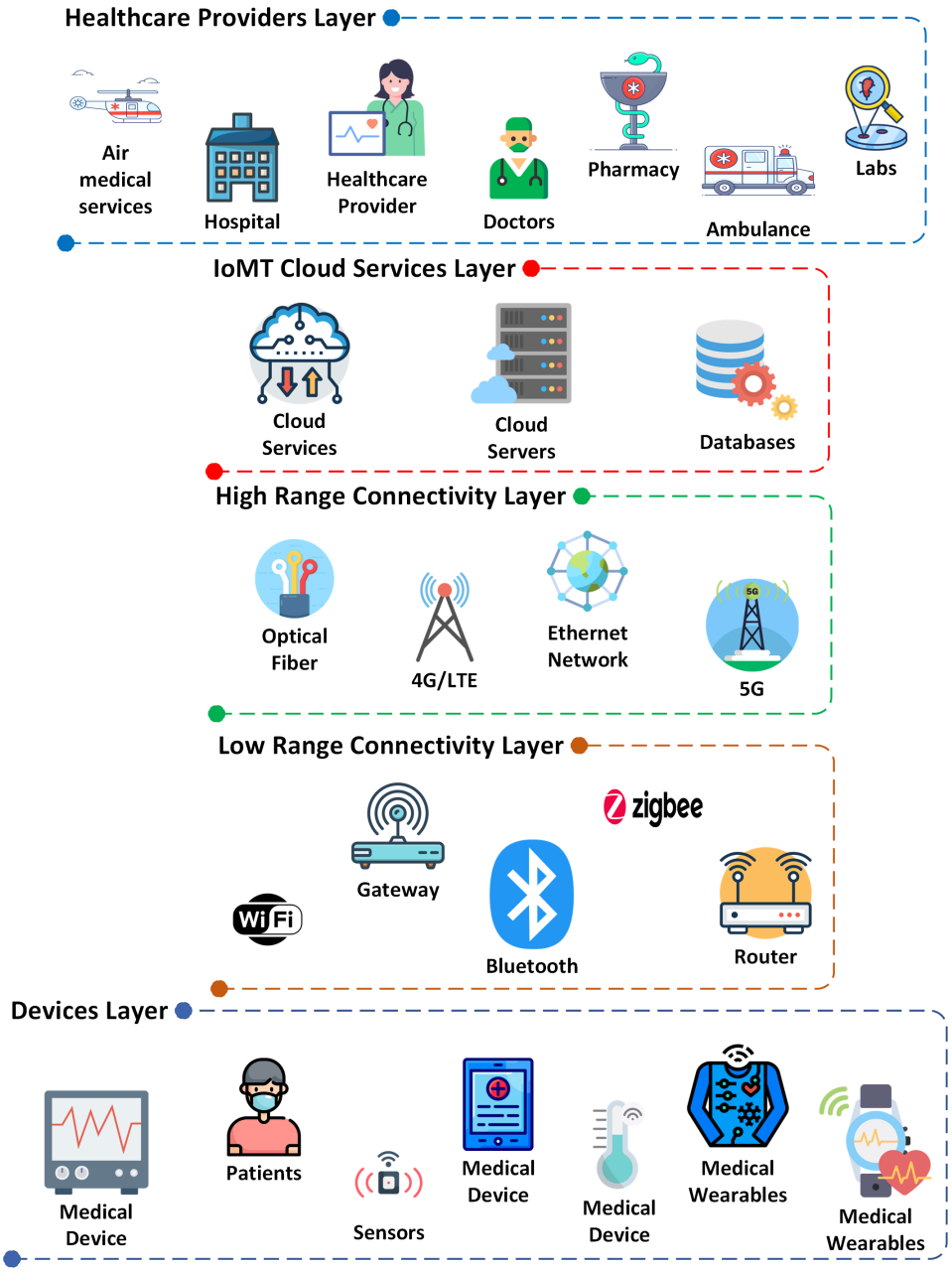

Figure 6.1: IoMT Architecture

rely on networked digital infrastructure as well as medical equipment and devices that use IoT sensors to connect them to centralized hospital networks, making it easy for

data sharing to facilitate and accelerate patient care. Since hospitals store an incredible amount of patient data, they must protect their patient records. The nature of patient data is highly confidential and should allow access to staff on-site and remotely on multiple devices. Confidential patient data can fetch much money for hackers, making the health industry a growing target for ransomware. Therefore, a major issue arises because this data is being sent through the Internet, a wireless public network susceptible to intruder attacks. The need for an IDS is crucial for network security in IoMT as it can detect and notify system administrators when a malicious attack or network intrusion occurs. IDSs can detect both insider attacks and external attacks that breach through existing security measures. Figure 6.1 shows how IoMT architecture can connect people (patients, users, and professional healthcare providers), IoMT devices (medical devices, wearable, and sensors), and patients' data through the communication technologies using the cloud/edge approach with aims to provide healthcare services to the system users.

Machine learning (ML) has the promise to provide an effective solution when it comes to misuse and anomaly detection. Two machine learning techniques that have attracted much attention in recent years in the field of deep learning are transfer and federated learning. Transfer learning is a technique that reuses pre-trained models with less data for solving new problems. Federated learning, on the other hand, is a technique that trains an algorithm across multiple decentralized platforms holding local data samples without exchanging them, making it an ideal candidate for use in medical cyber-physical systems. The main contributions of this chapter are in designing a novel federated transfer model that can build a scalable, customized global model out of local models to preserve the local models' privacy and lower training time and memory usage. Our proposed federated transfer learning-based IDS can secure the patient's healthcare-connected devices. Our model uses a DNN algorithm for training the network and transferring the knowledge

from the connected local edge models to build a global customized model without exposing data privacy. We demonstrate through simulation using the CICIDS2017 dataset that the performance of the proposed model is similar to the distributed learning approaches in terms of convergence and accuracy.

6.2 Federated Learning-based IDS (Use case: Internet of Medical Things (IoMT))

6.2.1 Model Workflow

Figure 6.2 shows the proposed model flowchart. The model works as follows:

1. Trains the cloud model with the global dataset (Source domain). By experiments, we found that considering the CICIDS2017-Friday dataset with CICIDS2017-Monday gives the best performance results when used as a source domain, thanks to the fact that the CICIDS2017-Friday dataset has the most number of attacks and instances among all the other dataset days. The CICIDS2017-Monday has only normal traffic, which will make the model learn and distinguish the normal traffic from the malicious traffic in a better way.

2. Distributes the pre-trained model results from the previous step to each of the connected gateways on the edge side. Each of the connected edge gateways can use its local dataset as shown in Figure 6.2 and feed it to the pre-trained cloud model to initiate the local model. The results of this step are discussed later in Section 6.3 (Performance Evaluation) in this chapter.

3. The newly trained edge models transfer back to the cloud side, where they are aggregated with the global cloud model using the parameters transferred from the

local edges. In our proposed model, the CICIDS2017 dataset shares the feature spaces and different instances for different source and target task labels (attacks).

4. Each device then utilizes the aggregated cloud model (global model) and its local dataset to train a customized model using federated learning, thus preserving data privacy while achieving the best possible performance (as close as possible to the performance achieved when directly training the model). The results of this stage of learning are detailed later in Section 6.3 of this chapter.

Algorithm 5: Federated Transfer Learning Procedure

Input: D_g: CICIDS2017-Monday ∪ CICIDS2017-Friday
Result: M_c: Customized Model for each IoMD IDS
Initialization: Train the DNN model M_g with the dataset D_g on the cloud side.
Distribution: M_g is distributed to I_L, all IoMT IDSs on the edge side.
for $k \leftarrow 1$ *to* N *(= Number of IoMT IDSs* I_L*)* **do**
    Transfer learning: I_L trains the distributed model M_g using local datasets D_L.
    Federated learning: I_L uploads the local trained model M_L parameters to the cloud side which is aggregated with the M_g.
end for
Transfer learning: Create a customized model M_c from the aggregated model for each local dataset.

6.2.2 Learning Process

In this proposed model, the FTL has been applied, and a pre-trained DNN model has been used to transfer the knowledge from the local models and build a global model without exposing data privacy as described in Algorithm 5. Since the knowledge of two or more datasets can be transferred with the neural networks using the features representation that maps the original features space into a shared feature subspace of those datasets, it will help deal with insufficient data instances and reduce the training time.

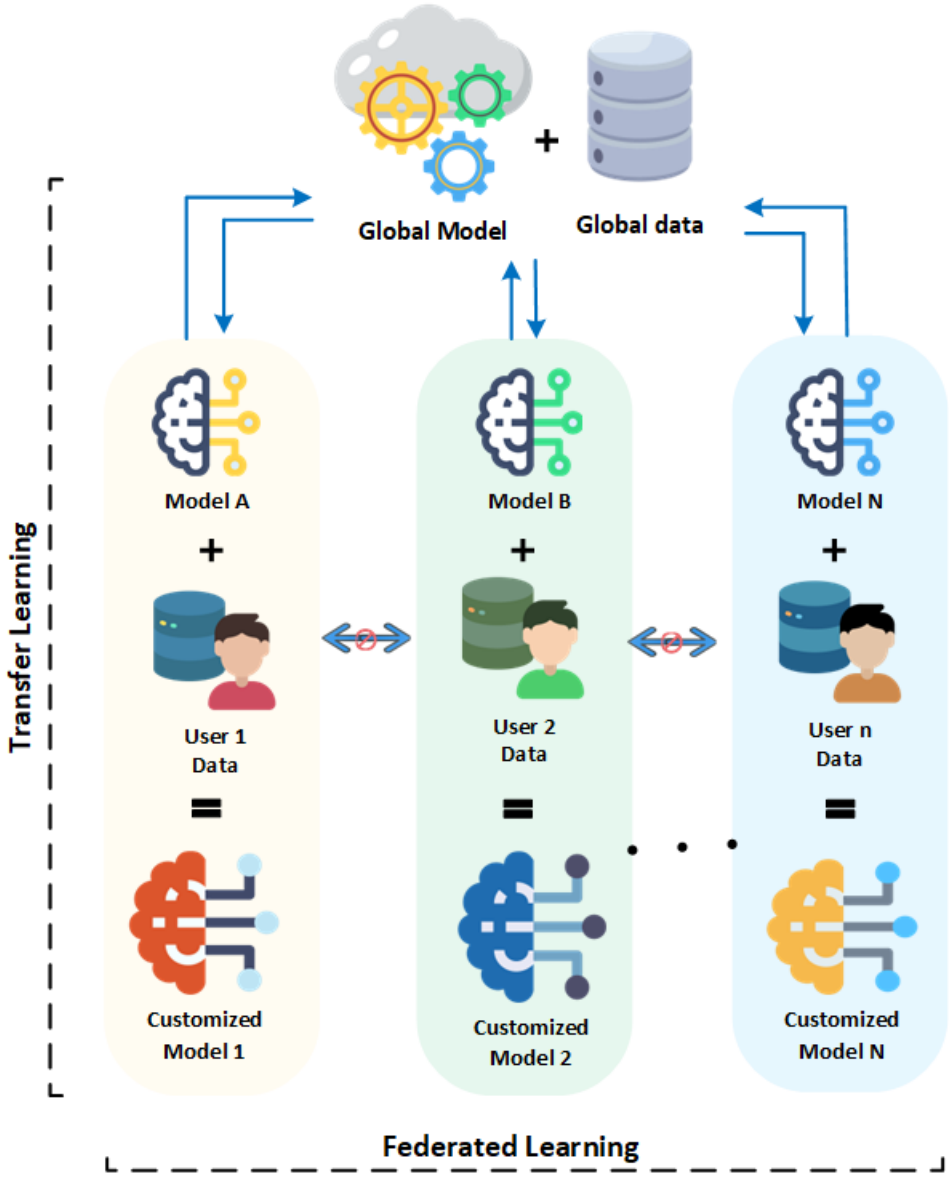

Figure 6.2: IoMT Model Flowchart

Therefore, the more relevant are the domains to apply the TL, the better model performance will be. The DNN in the proposed model consists of six neural network layers: one input, four hidden layers of 32, 16, 8, and 4 nodes, respectively, and one Softmax layer used for the classification purpose. The other parameter values and details used in

the network are shown in Table 6.1. The cloud model is trained with the global dataset (CICIDS2017-Monday + Friday) using the DNN algorithm as described previously in the model workflow, which will then be distributed to the other local edge models that will be trained with the edge dataset to minimize the loss function using the cross-entropy loss in both cases based on the following expression:

$$L(\Theta) = -\sum_{i=1}^{k} y_i \log\left(\hat{y}_i\right) \tag{6.1}$$

where i is the number of input training samples, and y is the label value. After all, the local models are trained with the global model and local data, and the parameters of those trained models will be uploaded to the cloud side to make a new aggregated model without sharing the user data as followed in [125]. The new aggregated model is the result of averaging the parameters used in the local models, and it is represented as follows:

$$M_g(w) = \frac{1}{K}\sum_{k=1}^{K} M_{lk}(w) \tag{6.2}$$

where K is the number of the local models, and w is the set of the parameters. This aggregated model can be applied directly to the edge datasets. However, it will perform poorly since this model learns only the common features between all the edge datasets and does not learn the customized features for each of the edge datasets. In this case, we need to transfer the learning on the edge sides by freezing all the layers initially, replacing the output layer with a new output layer, and starting to unfreeze the layers one by one for fine-tuning and getting a satisfactory performance. The model will freeze the first few layers with the low-level features and update the parameters of the other layers in the backpropagation process to make the model learn the customized features of each one of the edge datasets. In order to do that, a new alignment layer can be added to the

model to minimize the CORAL loss, which is computed as follows:

$$\ell_{CORAL} = \frac{1}{4d^2} \|C_g - C_l\|_F^2 \tag{6.3}$$

where $\|.\|_F^2$ represents the Euclidean norm, and d is the embedding feature dimension, C_g and the C_l are the covariance matrices of the global and the local parameters, respectively. The loss function of the local model can be calculated as follows:

$$arg\min_{\Theta_l} \ell_l = \sum_{i=1}^{n} \ell(f_l(X_i), y_i) + \sum_{i=1}^{n_l} \ell(f_l(X_i^l), y_i^l) + \lambda\, \ell_{CORAL} \tag{6.4}$$

where λ is the trade-off parameter, in this way, by adding a new edge dataset, the global model can generalize better without affecting the edge data privacy. Moreover, the global model can update itself by aligning with the edge models and distribute the updates to all edges for deploying transfer learning and obtaining a customized local model.

Table 6.1: Deep Neural Network Parameters

Parameter	**Value**
No. of Hidden Layers	4
No. of Nodes	32, 16, 8, 4
Optimizer	Adam
No. of Epochs	5
Loss (Objective function)	Binary Cross Entropy
Batch size	32
Activation Function	Relu, Sigmoid
Class Weight	None
Testing Size	33%

Table 6.2: Accuracy and time comparison for different scenarios using DNN

Source Domain	**Average Training Time (S)**	**Target Domain**	**Accuracy%**	**Avg. Prediction Time (S)**
CICIDS2017-Tues + CICIDS2017-Mon (Scenario 1)	120.8	CICIDS2017-Fri	71.16	16.7
		CICIDS2017-Wen	62.44	21.9
		CICIDS2017-Thur	99.51	13.9
CICIDS2017-Thur + CICIDS2017-Mon (Scenario 2)	127.0	CICIDS2017-Fri	71.16	17.7
		CICIDS2017-Tues	96.89	13.8
		CICIDS2017-Wen	63.46	20.8
CICIDS2017-Wed + CICIDS2017-Mon (Scenario 3)	133.5	CICIDS2017-Fri	69.47	16.9
		CICIDS2017-Tues	85.55	12.2
		CICIDS2017-Thur	84.06	14.0
CICIDS2017-Fri + CICIDS2017-Mon (Scenario 4)	227.1	CICIDS2017-Tues	95.14	10.3
		CICIDS2017-Wed	62.51	15.2
		CICIDS2017-Thur	88.27	12.1

6.3 Performance Evaluation

To evaluate the performance of the proposed model, we used the CICIDS2017 dataset (Described in Section 5.2.1). To verify the success of the federated transfer learning process, we considered accuracy, detection rate, average training and prediction time for the evaluation. The results of the comparison are shown in Tables 6.2 and 6.3. In Table 6.2, we simulated four scenario's as discussed in Section 6.2.1. A (Model Workflow); each one of the connected edges can use its local dataset to initiate its local model and upload it to the cloud model. The experiments show that combing CICIDS2017-Friday

Table 6.3: Accuracy and time comparison for different models in source domain (CICIDS2017-Fri + CICIDS2017-Mon)

Model	**Average Training Time (S)**	**Target Domain**	**Accuracy%**	**Average Prediction Time (S)**
Proposed Model	227.1	CICIDS2017-Tues	95.14	10.3
		CICIDS2017-Wed	62.51	15.2
		CICIDS2017-Thur	88.27	12.1
DBN	11140.1	CICIDS2017-Tues	92.05	25.2
		CICIDS2017-Wed	75.29	45.8
		CICIDS2017-Thur	81.78	27.1
DT	36.6	CICIDS2017-Tues	59.54	0.4
		CICIDS2017-Wed	62.03	0.5
		CICIDS2017-Thur	90.78	0.5
SGD	9.8	CICIDS2017-Tues	71.77	0.7
		CICIDS2017-Wed	56.24	1.0
		CICIDS2017-Thur	63.96	0.7
SVM	69397.9	CICIDS2017-Tues	72.80	3417.9
		CICIDS2017-Wed	57.12	4590.5
		CICIDS2017-Thur	64.51	3725.8

with CICIDS2017-Monday datasets to train the global model gives the best performance results; due to the presence of the largest number of attacks and instances in CICIDS2017-Friday, and the pure benign traffic data in CICIDS2017-Monday, which will make the model learns and distinguish the normal from the malicious traffic in a better way. We verified in Table 6.2 that the proposed model with the chosen datasets had the best accuracy values when the other local datasets used it to train in the first learning stage (Model Workflow Section 6.2.1 - Step 1). In Table 6.3, we compared the accuracy, aver-

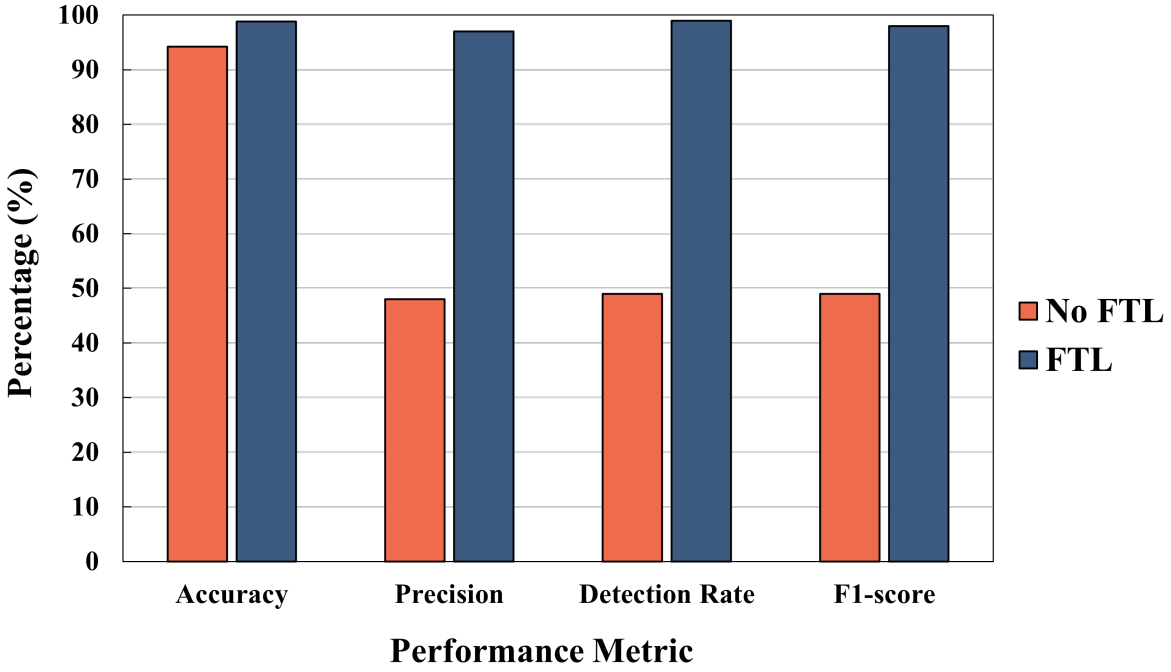

Figure 6.3: Customized Model on CICIDS2017-Tuesday

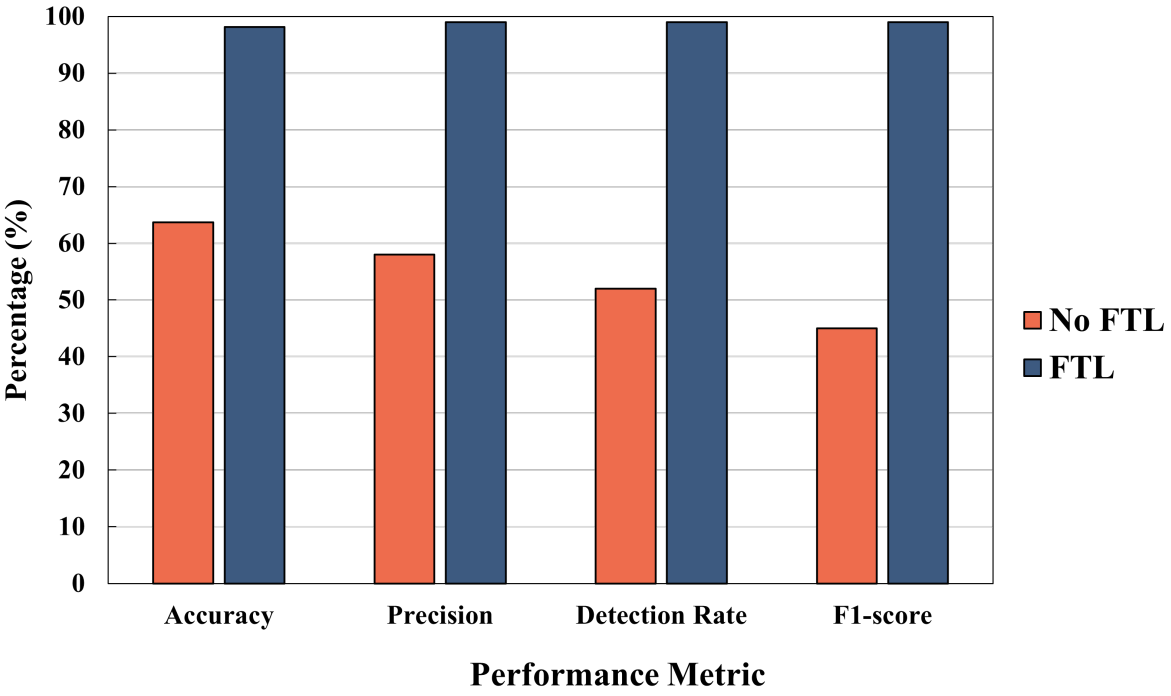

Figure 6.4: Customized Model on CICIDS2017-Wednesday

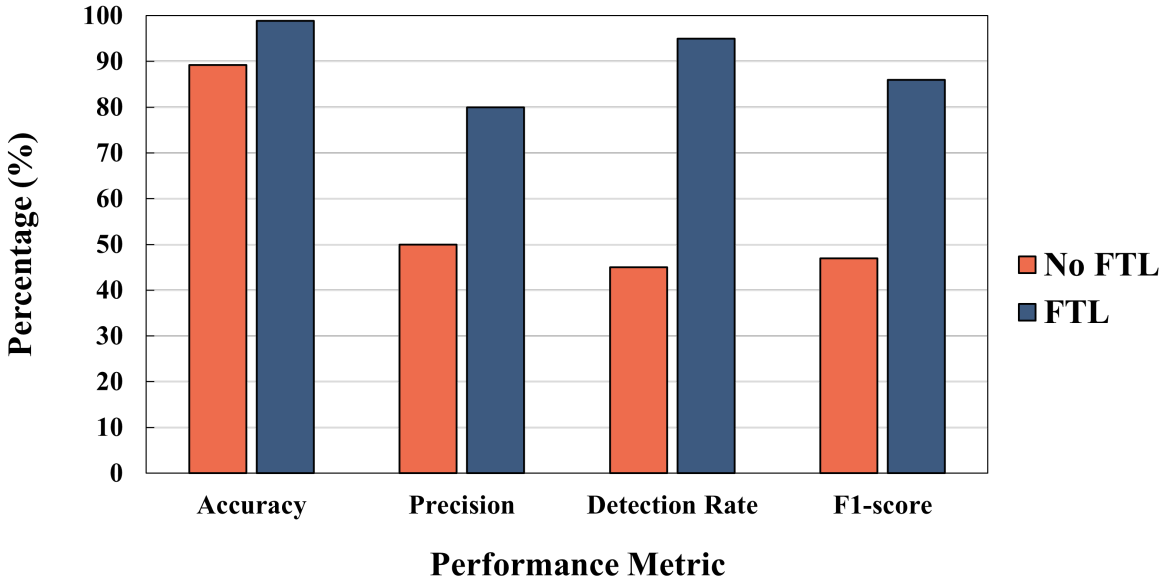

Figure 6.5: Customized Model on CICIDS2017-Thursday

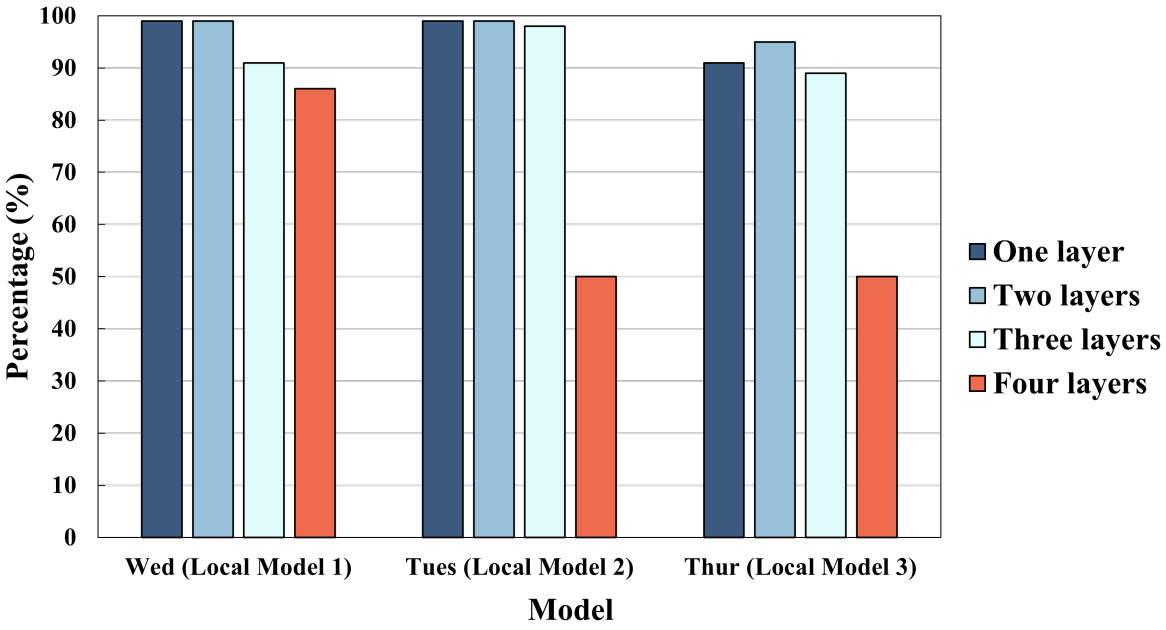

Figure 6.6: Detection Rate for Different Frozen Layers

age training and prediction times of the proposed model with four well-known ML/DL models that are used as traditional centralized learning methods, namely DBN, DT, Stochastic Gradient Descent (SGD), and SVM.

The results indicate that the proposed model has the best accuracy among the other algorithms. However, we can also see that even the training time of our proposed model is higher than some of the other standard ML algorithms, such as DT and SGD. However, it has a fair prediction time among the other algorithms. This is one of the federated learning features that offers real-time predictions on edge devices with the availability of local data without the need to access the cloud servers. In the Figures 6.3, 6.4, and 6.5, we analyzed the four well-known performance metrics: accuracy, precision, recall, and F1-score for the customized models 1, 2, and 3 when used to predict each one of the local datasets CICIDS2017 Tuesday, Wednesday, and Thursday, respectively. The results show that the proposed model achieved a better performance in the second transfer learning stage discussed in the model workflow (Section 6.2.1 - Step 4), as the customized models learnt the high-level features of each one of the local datasets; this indicates that our proposed model has the ability to learn incrementally from the newly attached local user datasets. We can also see a drop in the precision value of Model-3 (Figure 6.5); this is because the Thursday dataset is the most imbalanced dataset among other datasets where we have the most instances of the normal traffic. In Figure 6.6, we showed the relationship between the number of frozen layers and the change in the detection rate for each of the three customized models. The experiments show that freezing the first two hidden layers and updating the parameters in the other two hidden layers in backpropagation gives the best detection rate for all the customized models, which is due to the fact that the last two layers have the high-level features specified for each one of the local datasets. All the experiments are done using Intel Core i7 CPU-based workstation with 16GB DDR4 RAM, and the algorithms are implemented using Keras/Pytorch running on top of the TensorFlow Python library.

6.4 Summary

This chapter proposed a Federated Transfer Learning-based IDS to secure the IoMT networks. The model uses DNN algorithm to collaborate training the cloud model by the edge models and use the CICIDS2017 dataset to evaluate the accuracy and detection rate performance metrics. The results show the effectiveness of the proposed model in generalizing and learning incrementally compared with other baseline ML/DL algorithms used in the traditional centralized learning methods.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

Due to the growing number of IoT based networks, this thesis presents a novel, deep-learning based IDS to identify severe anomalies. In Chapter 2, we reviewed significant ML and DL approaches to achieve secure IoT environments. We first presented a brief overview of IoT architecture, IoT security issues, and some of ML and DL approaches, then we discussed and compared state-of-the art ML and DL methods such as CNN, RNN, LSTM, AE, RBM, DBN and DRL. We also critically evaluated important recent work in the IoT security field, explained the role of various ML and DL security methods and identified major problems with the state-of-the-art approaches. Our extensive examination also highlighted research problems and gaps that exist in current IoT security schemes.

Chapter 3 proposed the DL-IDS, in which a SMO algorithm is used to extract the most relevant features from the dataset. A SDPN is then applied to identify the optimal features and classify the data as normal or anomalous in different attack categories (e.g., DoS, U2R, R2L, and probe). We evaluated our DL-IDS system using the NSL-KDD

dataset, and demonstrated its superior performance with respect to accuracy (99.02%), precision (99.38%), recall (98.91%), and F1-score (99.14%).

In Chapter 4, we proposed a model that combines signature and anomaly-based IDS. The three phases considered in this model are traffic filtering, preprocessing and hybrid. In the traffic filtering phase, the features of the arrived packet streams are extracted and validated by the IoT gateway. In the preprocessing phase, the features are converted into numeric values, then normalized and redundancy reduced. Preprocessing concentrates on the network traffic and data previously collected from the dataset. The traffic packets then enter the hybrid IDS phase, where the signature-based IDS is applied using signature matching and the LightNet algorithm. All unknown packets are processed by the anomaly-based IDS, and the deep Q-learning algorithm then considers the SNR and bandwidth for attack classification. The proposed AS-IDS model shows greater improvement over the existing IDS methods.

In Chapter 5, we used the concept of transfer learning to transfer the knowledge between two different benchmark datasets datasets; CICIDS2017 and CSE-CIC-IDS2018. By applying two different deep learning algorithms, namely Deep Neural Network (DNN) and Convolutional Neural Network (CNN), we were able to achieve satisfying performance and reduced training/fine-tuning time for the target domains. In the second half of this chapter, we proposed an infrastructure-independent IDS to secure the intra-vehicle and external networks. The anomaly-based IDS is augmented with a blacklist of attacks signatures that are placed in the connected vehicles and uses an over-the-air-update concept to update the signatures of the attacks from a cloud database where the signatures of the new attacks can be generated and distributed to the connected vehicles. The detection engine is used the STL to transfer the knowledge of a pre-trained DBN model from the source domain and construct a feed-forward DNN where RF algorithm is used

to select the significant features. The CICIDS2017 and NSL-KDD datasets were used to evaluate the performance metrics accuracy, precision, detection rate, F1-score, and ROC curves.

Chapter 6 proposed a Federated Transfer Learning-based IDS to secure the IoMT networks. The model used DNN algorithm to collaborate training the cloud model by the edge models and used the CICIDS2017 dataset to evaluate the performance metrics such as the accuracy and detection rate. The results show the effectiveness of the proposed model in generalization and step-by-step learning compared to other basic ML/DL algorithms used in traditional intensive learning methods.

7.2 Future Work

In the future, this work can be extended by enabling blockchain technology to provide a high level of security and privacy for gaining training on sensitive and critical information. A blockchain is a digital ledger of transactions that are replicated and distributed throughout the whole network of computers on the blockchain. A block in a chain contains several transactions, and a record of every transaction is added to every participant's ledger when a new transaction is recorded on the blockchain. This distributed database is known as Distributed Ledger Technology (DLT). Blockchain is often used in domains such as healthcare, supply chain management, and the IoT since it provides strong protection against data tampering, locking access to devices, and allowing compromised devices to be shut down. It also can be used to create secured mesh network that will allow IoT devices to connect securely and reliably avoiding the threats of device spoofing and impersonation. On the other hand, despite the benefits of federated learning and transfer learning techniques, such as improving model performance, reducing training time, and maintaining data privacy, they can present a few security challenges,

such as server trust issues for model aggregation without the need of a trusted third party, as well as the server can be considered a single point of failure if attacked, negatively impacting the entire system [132]. The models used federated learning can be extended by enabling the blockchain technology to provide a high level of security and privacy for gaining training on sensitive and critical information, as well as analyzing sensitive data; applying blockchain technology to the FTL technique will ensure a high capacity of data analytics for collaborative learning with minimum response time, cost, and solve the issues mentioned above.

References

[1] M. Hasan, "State of iot 2022: Number of connected iot devices growing 18% to 14.4 billion globally," tech. rep., IoT Analytics, https://iot-analytics.com/number-connected-iot-devices/, 18 May, 2022.

[2] D. E. Kouicem, A. Bouabdallah, and H. Lakhlef, "Internet of things security: A top-down survey," *Computer Networks*, vol. 141, pp. 199–221, 2018.

[3] F. S. d. Lima Filho, F. A. Silveira, A. de Medeiros Brito Junior, G. Vargas-Solar, and L. F. Silveira, "Smart detection: an online approach for dos/ddos attack detection using machine learning," *Security and Communication Networks*, vol. 2019, 2019.

[4] K. Yang, J. Ren, Y. Zhu, and W. Zhang, "Active learning for wireless iot intrusion detection," *IEEE Wireless Communications*, vol. 25, no. 6, pp. 19–25, 2018.

[5] R. Ahsan, W. Shi, and J.-P. Corriveau, "Network intrusion detection using machine learning approaches: Addressing data imbalance," *IET Cyber-Physical Systems: Theory & Applications*, vol. 7, no. 1, pp. 30–39, 2022.

[6] Y. Otoum and A. Nayak, "On securing iot from deep learning perspective," in *2020 IEEE Symposium on Computers and Communications (ISCC)*, pp. 1–7, IEEE, 2020.

[7] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, "Machine learning for internet of things data analysis: A survey," *Digital Communications and Networks*, vol. 4, no. 3, pp. 161–175, 2018.

[8] D. Wang, D. Chen, B. Song, N. Guizani, X. Yu, and X. Du, "From iot to 5g i-iot: The next generation iot-based intelligent algorithms and 5g technologies," *IEEE Communications Magazine*, vol. 56, no. 10, pp. 114–120, 2018.

[9] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1125–1142, 2017.

[10] T. Qiu, N. Chen, K. Li, M. Atiquzzaman, and W. Zhao, "How can heterogeneous internet of things build our future: A survey," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2011–2027, 2018.

[11] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang, "A survey on the edge computing for the internet of things," *IEEE access*, vol. 6, pp. 6900–6919, 2017.

[12] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE communications surveys & tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[13] P. P. Ray, "A survey on internet of things architectures," *Journal of King Saud University-Computer and Information Sciences*, vol. 30, no. 3, pp. 291–319, 2018.

[14] V. Adat and B. Gupta, "Security in internet of things: issues, challenges, taxonomy, and architecture," *Telecommunication Systems*, vol. 67, no. 3, pp. 423–441, 2018.

[15] R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning ddos detection for consumer internet of things devices," in *2018 IEEE Security and Privacy Workshops (SPW)*, pp. 29–35, IEEE, 2018.

[16] A. R. Sfar, E. Natalizio, Y. Challal, and Z. Chtourou, "A roadmap for security challenges in the internet of things," *Digital Communications and Networks*, vol. 4, no. 2, pp. 118–137, 2018.

[17] T. A. Ahanger and A. Aljumah, "Internet of things: A comprehensive study of security issues and defense mechanisms," *IEEE Access*, vol. 7, pp. 11020–11028, 2018.

[18] Y. Chen, S. Das, P. Dhar, A. El-Saddik, and A. Nayak, "Detecting and preventing ip-spoofed distributed dos attacks.," *IJ Network Security*, vol. 7, no. 1, pp. 69–80, 2008.

[19] A. Canziani, A. Paszke, and E. Culurciello, "An analysis of deep neural network models for practical applications," *arXiv preprint arXiv:1605.07678*, 2016.

[20] H. Li, K. Ota, and M. Dong, "Learning iot in edge: Deep learning for the internet of things with edge computing," *IEEE network*, vol. 32, no. 1, pp. 96–101, 2018.

[21] N. D. Lane, S. Bhattacharya, P. Georgiev, C. Forlivesi, and F. Kawsar, "An early resource characterization of deep learning on wearables, smartphones and internet-of-things devices," in *Proceedings of the 2015 international workshop on internet of things towards applications*, pp. 7–12, 2015.

[22] S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M. P. Reyes, M.-L. Shyu, S.-C. Chen, and S. Iyengar, "A survey on deep learning: Algorithms, techniques, and applications," *ACM Computing Surveys (CSUR)*, vol. 51, no. 5, pp. 1–36, 2018.

[23] R. F. Molanes, K. Amarasinghe, J. Rodriguez-Andina, and M. Manic, "Deep learning and reconfigurable platforms in the internet of things: challenges and opportunities in algorithms and hardware," *IEEE Industrial Electronics Magazine*, vol. 12, no. 2, pp. 36–49, 2018.

[24] M. Mohammadi, A. Al-Fuqaha, M. Guizani, and J.-S. Oh, "Semisupervised deep reinforcement learning in support of iot and smart city services," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 624–635, 2017.

[25] S. K. Lo, Y. Liu, Q. Lu, C. Wang, X. Xu, H.-Y. Paik, and L. Zhu, "Towards trustworthy ai: Blockchain-based architecture design for accountability and fairness of federated learning systems," *IEEE Internet of Things Journal*, 2022.

[26] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[27] Z. Teimoori, A. Yassine, and M. S. Hossain, "A secure cloudlet-based charging station recommendation for electric vehicles empowered by federated learning," *IEEE Transactions on Industrial Informatics*, 2022.

[28] A. Ferdowsi and W. Saad, "Deep learning for signal authentication and security in massive internet-of-things systems," *IEEE Transactions on Communications*, vol. 67, no. 2, pp. 1371–1387, 2018.

[29] R. Das, A. Gadre, S. Zhang, S. Kumar, and J. M. Moura, "A deep learning approach to iot authentication," in *2018 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2018.

[30] B. A. Tama and K.-H. Rhee, "Attack classification analysis of iot network via deep learning approach," *Res. Briefs Inf. Commun. Technol. Evol.(ReBICTE)*, vol. 3, pp. 1–9, 2017.

[31] A. A. Diro and N. Chilamkurti, "Distributed attack detection scheme using deep learning approach for internet of things," *Future Generation Computer Systems*, vol. 82, pp. 761–768, 2018.

[32] O. Brun, Y. Yin, E. Gelenbe, Y. M. Kadioglu, J. Augusto-Gonzalez, and M. Ramos, "Deep learning with dense random neural networks for detecting attacks against iot-connected home environments," in *International ISCIS Security Workshop*, pp. 79–89, Springer, Cham, 2018.

[33] P. M. Shakeel, S. Baskar, V. S. Dhulipala, S. Mishra, and M. M. Jaber, "Maintaining security and privacy in health care system using learning based deep-q-networks," *Journal of medical systems*, vol. 42, no. 10, p. 186, 2018.

[34] A. Azmoodeh, A. Dehghantanha, and K.-K. R. Choo, "Robust malware detection for internet of (battlefield) things devices using deep eigenspace learning," *IEEE Transactions on Sustainable Computing*, vol. 4, no. 1, pp. 88–95, 2018.

[35] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-baiot—network-based detection of iot botnet attacks using deep autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, 2018.

[36] C. D. McDermott, F. Majdani, and A. V. Petrovski, "Botnet detection in the internet of things using deep learning approaches," in *2018 international joint conference on neural networks (IJCNN)*, pp. 1–8, IEEE, 2018.

[37] A. Abeshu and N. Chilamkurti, "Deep learning: the frontier for distributed attack detection in fog-to-things computing," *IEEE Communications Magazine*, vol. 56, no. 2, pp. 169–175, 2018.

[38] A.-H. Muna, N. Moustafa, and E. Sitnikova, "Identification of malicious activities in industrial internet of things based on deep learning models," *Journal of Information Security and Applications*, vol. 41, pp. 1–11, 2018.

[39] F. Y. Yavuz, Ü. Devrim, and G. Ensar, "Deep learning for detection of routing attacks in the internet of things," *International Journal of Computational Intelligence Systems*, vol. 12, no. 1, pp. 39–58, 2018.

[40] A. Ferdowsi and W. Saad, "Deep learning-based dynamic watermarking for secure signal authentication in the internet of things," in *2018 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2018.

[41] G. Thamilarasu and S. Chawla, "Towards deep-learning-driven intrusion detection for the internet of things," *Sensors*, vol. 19, no. 9, p. 1977, 2019.

[42] B. Arrington, L. Barnett, R. Rufus, and A. Esterline, "Behavioral modeling intrusion detection system (bmids) using internet of things (iot) behavior-based anomaly detection via immunity-inspired algorithms," in *2016 25th International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–6, IEEE, 2016.

[43] A. Alghuried, "A model for anomalies detection in internet of things (iot) using inverse weight clustering and decision tree," 2017.

[44] H. HaddadPajouh, A. Dehghantanha, R. Khayami, and K.-K. R. Choo, "A deep recurrent neural network based approach for internet of things malware threat hunting," *Future Generation Computer Systems*, vol. 85, pp. 88–96, 2018.

[45] Z. A. Khan and U. Abbasi, "Reputation management using honeypots for intrusion detection in the internet of things," *Electronics*, vol. 9, no. 3, p. 415, 2020.

[46] A. Khraisat, I. Gondal, P. Vamplew, J. Kamruzzaman, and A. Alazab, "A novel ensemble of hybrid intrusion detection system for detecting internet of things attacks," *Electronics*, vol. 8, no. 11, p. 1210, 2019.

[47] A. N. Iman and T. Ahmad, "Improving intrusion detection system by estimating parameters of random forest in boruta," in *2020 International Conference on Smart Technology and Applications (ICoSTA)*, pp. 1–6, IEEE, 2020.

[48] S. Rajagopal, P. P. Kundapur, and K. S. Hareesha, "A stacking ensemble for network intrusion detection using heterogeneous datasets," *Security and Communication Networks*, vol. 2020, 2020.

[49] Y. Y. Aung and M. M. Min, "Hybrid intrusion detection system using k-means and k-nearest neighbors algorithms," in *2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS)*, pp. 34–38, IEEE, 2018.

[50] L. Lv, W. Wang, Z. Zhang, and X. Liu, "A novel intrusion detection system based on an optimal hybrid kernel extreme learning machine," *Knowledge-Based Systems*, p. 105648, 2020.

[51] H. Alazzam, A. Sharieh, and K. E. Sabri, "A feature selection algorithm for intrusion detection system based on pigeon inspired optimizer," *Expert Systems with Applications*, vol. 148, p. 113249, 2020.

[52] M. Mazini, B. Shirazi, and I. Mahdavi, "Anomaly network-based intrusion detection system using a reliable hybrid artificial bee colony and adaboost algo-

rithms," *Journal of King Saud University-Computer and Information Sciences*, vol. 31, no. 4, pp. 541–553, 2019.

[53] Y. Zhang, P. Li, and X. Wang, "Intrusion detection for iot based on improved genetic algorithm and deep belief network," *IEEE Access*, vol. 7, pp. 31711–31722, 2019.

[54] F. Hachmi, K. Boujenfa, and M. Limam, "Enhancing the accuracy of intrusion detection systems by reducing the rates of false positives and false negatives through multi-objective optimization," *Journal of Network and Systems Management*, vol. 27, no. 1, pp. 93–120, 2019.

[55] E. Aminanto and K. Kim, "Deep learning in intrusion detection system: An overview," in *2016 International Research Conference on Engineering and Technology (2016 IRCET)*, Higher Education Forum, 2016.

[56] N. Balakrishnan, A. Rajendran, D. Pelusi, and V. Ponnusamy, "Deep belief network enhanced intrusion detection system to prevent security breach in the internet of things," *Internet of Things*, p. 100112, 2019.

[57] M. A. Khan, M. Karim, Y. Kim, *et al.*, "A scalable and hybrid intrusion detection system based on the convolutional-lstm network," *Symmetry*, vol. 11, no. 4, p. 583, 2019.

[58] Y. Liu, F. R. Yu, X. Li, H. Ji, and V. C. M. Leung, "Blockchain and machine learning for communications and networking systems," *IEEE Communications Surveys Tutorials*, vol. 22, no. 2, pp. 1392–1431, 2020.

[59] A. Qayyum, M. Usama, J. Qadir, and A. Al-Fuqaha, "Securing connected & autonomous vehicles: Challenges posed by adversarial machine learning and the way

forward," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 998–1026, 2020.

[60] Y. Otoum and A. Nayak, "Signature-over-the-air with transfer learning ids for intelligent connected vehicles (icv)," in *2021 IEEE Globecom Workshops (GC Wkshps)*, pp. 1–6, IEEE, 2021.

[61] Y. Otoum, Y. Wan, and A. Nayak, "Federated transfer learning-based ids for the internet of medical things (iomt)," in *2021 IEEE Globecom Workshops (GC Wkshps)*, pp. 1–6, IEEE, 2021.

[62] X. Li, Z. Hu, M. Xu, Y. Wang, and J. Ma, "Transfer learning based intrusion detection scheme for internet of vehicles," *Information Sciences*, vol. 547, pp. 119–135, 2021.

[63] S. Tariq, S. Lee, and S. S. Woo, "Cantransfer: transfer learning based intrusion detection on a controller area network using convolutional lstm network," in *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, pp. 1048–1055, 2020.

[64] F. Jin, M. Chen, W. Zhang, Y. Yuan, and S. Wang, "Intrusion detection on internet of vehicles via combining log-ratio oversampling, outlier detection and metric learning," *Information Sciences*, vol. 579, pp. 814–831, 2021.

[65] L. Vu, Q. U. Nguyen, D. N. Nguyen, D. T. Hoang, and E. Dutkiewicz, "Deep transfer learning for iot attack detection," *IEEE Access*, vol. 8, pp. 107335–107344, 2020.

[66] B. Sengupta and A. Lakshminarayanan, "Distritrust: Distributed and low-latency

access validation in zero-trust architecture," *Journal of Information Security and Applications*, vol. 63, p. 103023, 2021.

[67] R. Vanickis, P. Jacob, S. Dehghanzadeh, and B. Lee, "Access control policy enforcement for zero-trust-networking," in *2018 29th Irish Signals and Systems Conference (ISSC)*, pp. 1–6, IEEE, 2018.

[68] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020.

[69] X. Li, Y. Grandvalet, F. Davoine, J. Cheng, Y. Cui, H. Zhang, S. Belongie, Y.-H. Tsai, and M.-H. Yang, "Transfer learning in computer vision tasks: Remember where you come from," *Image and Vision Computing*, vol. 93, p. 103853, 2020.

[70] A. Jaiswal, N. Gianchandani, D. Singh, V. Kumar, and M. Kaur, "Classification of the covid-19 infected patients using densenet201 based deep transfer learning," *Journal of Biomolecular Structure and Dynamics*, pp. 1–8, 2020.

[71] Y. Liu and C. Xiao, "Transfer learning for hyperspectral image classification using convolutional neural network," in *MIPPR 2019: Remote Sensing Image Processing, Geographic Information Systems, and Other Applications* (Z. Cao, J. Ma, Z. Chen, and Y. Shi, eds.), vol. 11432, pp. 79 – 84, SPIE, 2020.

[72] S. Ruder, M. E. Peters, S. Swayamdipta, and T. Wolf, "Transfer learning in natural language processing," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*, (Minneapolis, Minnesota), pp. 15–18, Association for Computational Linguistics, June 2019.

[73] R. Zhao, Y. Yin, Y. Shi, and Z. Xue, "Intelligent intrusion detection based on federated learning aided long short-term memory," *Physical Communication*, vol. 42, p. 101157, 2020.

[74] N. A. A. Al-Marri, B. S. Ciftler, and M. M. Abdallah, "Federated mimic learning for privacy preserving intrusion detection," *CoRR*, vol. abs/2012.06974, 2020.

[75] B. Li, Y. Wu, J. Song, T. Li, and L. Zhao, "Deepfed: Federated deep learning for intrusion detection in industrial cyber-physical systems," *IEEE Transactions on Industrial Informatics*, vol. 7, pp. 5615–5624, 2020.

[76] Y. Sun, H. Ochiai, and H. Esaki, "Intrusion detection with segmented federated learning for large-scale multiple lans," in *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2020.

[77] Y. Zhao, J. Chen, D. Wu, J. Teng, and S. Yu, "Multi-task network anomaly detection using federated learning," *SoICT 2019: Proceedings of the Tenth International Symposium on Information and Communication Technology*, pp. 273–279, 2019.

[78] V. Mothukuri, P. Khare, R. M. Parizi, S. Pouriyeh, A. Dehghantanha, and G. Srivastava, "Federated learning-based anomaly detection for iot security attacks," *IEEE Internet of Things Journal*, 2021.

[79] D. Preuveneers, V. Rimmer, I. Tsingenopoulos, J. Spooren, W. Joosen, and E. Ilie-Zudor, "Chained anomaly detection models for federated learning: An intrusion detection case study," *Applied Sciences*, vol. 8, no. 12, 2018.

[80] Z. Yang, M. Chen, K. Wong, H. V. Poor, and S. Cui, "Federated learning for 6g: Applications, challenges, and opportunities," *CoRR*, vol. abs/2101.01338, 2021.

[81] A. A. Hady, A. Ghubaish, T. Salman, D. Unal, and R. Jain, "Intrusion detection system for healthcare systems using medical and network data: A comparison study," *IEEE Access*, vol. 8, pp. 106576–106584, 2020.

[82] S. Otoum, I. Al Ridhawi, and H. T. Mouftah, "Blockchain-supported federated learning for trustworthy vehicular networks," in *GLOBECOM 2020-2020 IEEE Global Communications Conference*, pp. 1–6, IEEE, 2020.

[83] Z. Du, C. Wu, T. Yoshinaga, K. Yau, Y. Ji, and J. Li, "Federated learning for vehicular internet of things: Recent advances and open issues," *IEEE Computer Graphics and Applications*, pp. 1–1, may 2020.

[84] L. Yang, A. Moubayed, I. Hamieh, and A. Shami, "Tree-based intelligent intrusion detection system in internet of vehicles," in *IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2019.

[85] H. M. Song, H. R. Kim, and H. K. Kim, "Intrusion detection system based on the analysis of time intervals of can messages for in-vehicle network," in *IEEE International Conference on Information Networking (ICOIN)*, pp. 63–68, 2016.

[86] D. Kosmanos, A. Pappas, F. J. Aparicio-Navarro, L. Maglaras, H. Janicke, E. Boiten, and A. Argyriou, "Intrusion detection system for platooning connected autonomous vehicles," in *4th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM)*, pp. 1–9, 2019.

[87] D. Kosmanos, A. Pappas, L. Maglaras, S. Moschoyiannis, F. J. Aparicio-Navarro, A. Argyriou, and H. Janicke, "A novel intrusion detection system against spoofing attacks in connected electric vehicles," *Array*, vol. 5, p. 100013, 2020.

[88] H. Lee, S. H. Jeong, and H. K. Kim, "Otids: A novel intrusion detection system for in-vehicle network by using remote frame," in *15th IEEE Annual Conference on Privacy, Security and Trust (PST)*, pp. 57–5709, 2017.

[89] J. Canedo and A. Skjellum, "Using machine learning to secure iot systems," in *2016 14th annual conference on privacy, security and trust (PST)*, pp. 219–222, IEEE, 2016.

[90] M. Asad, M. Asim, T. Javed, M. O. Beg, H. Mujtaba, and S. Abbas, "Deepdetect: detection of distributed denial of service attacks using deep learning," *The Computer Journal*, vol. 63, no. 7, pp. 983–994, 2020.

[91] Y. Otoum, D. Liu, and A. Nayak, "Dl-ids: a deep learning–based intrusion detection framework for securing iot," *Transactions on Emerging Telecommunications Technologies*, vol. 33, no. 3, p. e3803, 2022.

[92] J. C. Bansal, H. Sharma, S. S. Jadon, and M. Clerc, "Spider monkey optimization algorithm for numerical optimization," *Memetic computing*, vol. 6, no. 1, pp. 31–47, 2014.

[93] J. Shi, S. Zhou, X. Liu, Q. Zhang, M. Lu, and T. Wang, "Stacked deep polynomial network based representation learning for tumor classification with small ultrasound image dataset," *Neurocomputing*, vol. 194, pp. 87–94, 2016.

[94] X. Zheng, J. Shi, Y. Li, X. Liu, and Q. Zhang, "Multi-modality stacked deep polynomial network based feature learning for alzheimer's disease diagnosis," in *2016 IEEE 13th international symposium on biomedical imaging (ISBI)*, pp. 851–854, IEEE, 2016.

[95] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE symposium on computational intelligence for security and defense applications*, pp. 1–6, IEEE, 2009.

[96] Y. Zhou, M. Han, L. Liu, J. S. He, and Y. Wang, "Deep learning approach for cyberattack detection," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 262–267, IEEE, 2018.

[97] Y. Otoum and A. Nayak, "As-ids: Anomaly and signature based ids for the internet of things," *Journal of Network and Systems Management*, vol. 29, no. 3, pp. 1–26, 2021.

[98] S. Otoum, N. Guizani, and H. Mouftah, "On the feasibility of split learning, transfer learning and federated learning for preserving security in its systems," *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[99] Y. Otoum, Y. Wan, and A. Nayak, "Transfer learning-driven intrusion detection for internet of vehicles (iov)," in *2022 International Wireless Communications and Mobile Computing (IWCMC)*, pp. 342–347, IEEE, 2022.

[100] L. Santos, C. Rabadao, and R. Gonçalves, "Intrusion detection systems in internet of things: A literature review," in *2018 13th Iberian Conference on Information Systems and Technologies (CISTI)*, pp. 1–7, IEEE, 2018.

[101] Y. Fu, Z. Yan, J. Cao, O. Koné, and X. Cao, "An automata based intrusion detection method for internet of things," *Mobile Information Systems*, vol. 2017, 2017.

[102] M. F. Elrawy, A. I. Awad, and H. F. Hamed, "Intrusion detection systems for iot-

based smart environments: a survey," *Journal of Cloud Computing*, vol. 7, no. 1, p. 21, 2018.

[103] U. R. Salunkhe and S. N. Mali, "Security enrichment in intrusion detection system using classifier ensemble," *Journal of Electrical and Computer Engineering*, vol. 2017, 2017.

[104] K. Vengatesan, A. Kumar, R. Naik, and D. K. Verma, "Anomaly based novel intrusion detection system for network traffic reduction," in *2018 2nd International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC) I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC), 2018 2nd International Conference on*, pp. 688–690, IEEE, 2018.

[105] Ö. Cepheli, S. Büyükçorak, and G. Karabulut Kurt, "Hybrid intrusion detection system for ddos attacks," *Journal of Electrical and Computer Engineering*, vol. 2016, 2016.

[106] A. I. Saleh, F. M. Talaat, and L. M. Labib, "A hybrid intrusion detection system (hids) based on prioritized k-nearest neighbors and optimized svm classifiers," *Artificial Intelligence Review*, vol. 51, no. 3, pp. 403–443, 2019.

[107] A. Khraisat, I. Gondal, P. Vamplew, J. Kamruzzaman, and A. Alazab, "Hybrid intrusion detection system based on the stacking ensemble of c5 decision tree classifier and one class support vector machine," *Electronics*, vol. 9, no. 1, p. 173, 2020.

[108] I. A. Khan, D. Pi, Z. U. Khan, Y. Hussain, and A. Nawaz, "Hml-ids: A hybrid-multilevel anomaly prediction approach for intrusion detection in scada systems," *IEEE Access*, vol. 7, pp. 89507–89521, 2019.

[109] R. Elhefnawy, H. Abounaser, and A. Badr, "A hybrid nested genetic-fuzzy algorithm framework for intrusion detection and attacks," *IEEE Access*, 2020.

[110] K. Jiang, W. Wang, A. Wang, and H. Wu, "Network intrusion detection combined hybrid sampling with deep hierarchical network," *IEEE Access*, vol. 8, pp. 32464–32476, 2020.

[111] J. Kim, J. Kim, H. Kim, M. Shim, and E. Choi, "Cnn-based network intrusion detection against denial-of-service attacks," *Electronics*, vol. 9, no. 6, p. 916, 2020.

[112] A. M. Al Tobi and I. Duncan, "Improving intrusion detection model prediction by threshold adaptation," *Information*, vol. 10, no. 5, p. 159, 2019.

[113] R. Magán-Carrión, D. Urda, I. Díaz-Cano, and B. Dorronsoro, "Towards a reliable comparison and evaluation of network intrusion detection systems based on machine learning approaches," *Applied Sciences*, vol. 10, no. 5, p. 1775, 2020.

[114] Q. Ye and W. Zhi, "Discrete hessian eigenmaps method for dimensionality reduction," *Journal of Computational and Applied Mathematics*, vol. 278, pp. 197–212, 2015.

[115] Y. Tang and S. Chen, "An automated signature-based approach against polymorphic internet worms," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 7, pp. 879–892, 2007.

[116] A. H. Khan, "Lightweight neural networks," *arXiv preprint arXiv:1712.05695*, 2017.

[117] S. J. Mousavirad and H. Ebrahimpour-Komleh, "Human mental search: a new population-based metaheuristic optimization algorithm," *Applied Intelligence*, vol. 47, no. 3, pp. 850–887, 2017.

[118] S. Kaur and M. Singh, "Hybrid intrusion detection and signature generation using deep recurrent neural networks," *Neural Computing and Applications*, pp. 1–19, 2019.

[119] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.

[120] C. Bernardini, M. R. Asghar, and B. Crispo, "Security and privacy in vehicular communications: Challenges and opportunities," *Vehicular Communications*, vol. 10, pp. 13–28, 2017.

[121] L. Pan, X. Zheng, H. Chen, T. Luan, H. Bootwala, and L. Batten, "Cyber security attacks to modern vehicular systems," *Journal of information security and applications*, vol. 36, pp. 90–100, 2017.

[122] M. Dibaei, X. Zheng, K. Jiang, R. Abbas, S. Liu, Y. Zhang, Y. Xiang, and S. Yu, "Attacks and defences on intelligent connected vehicles: A survey," *Digital Communications and Networks*, 2020.

[123] R. Panigrahi and S. Borah, "A detailed analysis of cicids2017 dataset for designing intrusion detection systems," *International Journal of Engineering & Technology*, vol. 7, no. 3.24, pp. 479–482, 2018.

[124] J. L. Leevy and T. M. Khoshgoftaar, "A survey and analysis of intrusion detection models based on cse-cic-ids2018 big data," *Journal of Big Data*, vol. 7, no. 1, pp. 1–19, 2020.

[125] X. Li, H.-l. Chi, W. Lu, F. Xue, J. Zeng, and C. Z. Li, "Federated transfer learning enabled smart work packaging for preserving personal image information of construction worker," *Automation in Construction*, vol. 128, p. 103738, 2021.

[126] D. Xu, Y. Shi, I. W. Tsang, Y.-S. Ong, C. Gong, and X. Shen, "Survey on multi-output learning," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 7, pp. 2409–2429, 2019.

[127] M. T. Garip, M. E. Gursoy, P. Reiher, and M. Gerla, "Congestion attacks to autonomous cars using vehicular botnets," in *NDSS Workshop on Security of Emerging Networking Technologies (SENT), San Diego, CA*, 2015.

[128] F. Sagstetter, M. Lukasiewycz, S. Steinhorst, M. Wolf, A. Bouard, W. R. Harris, S. Jha, T. Peyrin, A. Poschmann, and S. Chakraborty, "Security challenges in automotive hardware/software architecture design," in *2013 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 458–463, IEEE, 2013.

[129] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization.," in *ICISSp*, pp. 108–116, 2018.

[130] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2021.

[131] Cybersecurity-Ventures, "Healthcare Industry To Spend $125 Billion On Cybersecurity From 2020 to 2025," 2021.

[132] D. C. Nguyen, M. Ding, Q.-V. Pham, P. N. Pathirana, L. B. Le, A. Seneviratne, J. Li, D. Niyato, and H. V. Poor, "Federated learning meets blockchain in edge computing: Opportunities and challenges," *IEEE Internet of Things Journal*, 2021.