Securing Connected and Automated Surveillance Systems against Network Intrusions and Adversarial Attacks

by

Abdul Jabbar Siddiqui

A thesis
submitted to the University of Ottawa in partial
fulfillment of the thesis requirement for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Abstract

In the recent years, connected surveillance systems have been witnessing an unprecedented evolution owing to the advancements in internet of things and deep learning technologies. However, vulnerabilities to various kinds of attacks both at the cyber network-level and at the physical world-level are also rising. This poses danger not only to the devices but also to human life and property. The goal of this thesis is to enhance the security of an internet of things, focusing on connected video-based surveillance systems, by proposing multiple novel solutions to address security issues at the cyber network-level and to defend such systems at the physical world-level.

In order to enhance security at the cyber network-level, this thesis designs and develops solutions to detect network intrusions in an internet of things such as surveillance cameras. The first solution is a novel method for network flow features transformation, named *TempoCode*. It introduces a temporal codebook-based encoding of flow features based on capturing the key patterns of benign traffic in a learnt temporal codebook. The second solution takes an unsupervised learning-based approach and proposes four methods to build efficient and adaptive ensembles of neural networks-based autoencoders for intrusion detection in internet of things such as surveillance cameras.

To address the physical world-level attacks, this thesis studies, for the first time to the best of our knowledge, adversarial patches-based attacks against a convolutional neural network (CNN)-based surveillance system designed for vehicle make and model recognition (VMMR). The connected video-based surveillance systems that are based on deep learning models such as CNNs are highly vulnerable to adversarial machine learning-based attacks that could trick and fool the surveillance systems. In addition, this thesis proposes and evaluates a lightweight defense solution called *SIHFR* to mitigate the impact of such adversarial-patches on CNN-based VMMR systems, leveraging the symmetry in vehicles' face images.

The experimental evaluations on recent realistic intrusion detection datasets prove the effectiveness of the developed solutions, in comparison to state-of-the-art, in detecting intrusions of various types and for different devices. Moreover, using a real-world surveillance dataset, we demonstrate the effectiveness of the *SIHFR* defense method which does not require re-training of the target VMMR model and adds only a minimal overhead. The solutions designed and developed in this thesis shall pave the way forward for future studies to develop efficient intrusion detection systems and adversarial attacks mitigation methods for connected surveillance systems such as VMMR.

List of Publications

The work accomplished through this thesis has resulted in the following journal and conference publications:

Journals

- Boukerche, A., Siddiqui, A.J., Mammeri, A. "Automated Vehicle Detection and Classification: Models, Methods, and Techniques," ACM Computing Surveys, Vol:50, Issue:5, Article 62 (November 2017)
- Siddiqui, A.J., Boukerche, A. "TempoCode-IoT: Temporal Codebook-based Encoding of Flow Features for Intrusion Detection in Internet of Things," Cluster Computing, Springer Nature (2020).
- Siddiqui, A.J., Boukerche, A. "Adaptive Ensembles of Autoencoders for Unsupervised IoT Network Intrusion Detection," Computing, Springer Nature (2021).
- Siddiqui, A.J., Boukerche, A. "SIHFR: A Lightweight Defense Method against Adversarial Patches-based Attacks on Automated Vehicle Make and Model Recognition Systems," Journal of Network and Systems Management, Springer Nature (2021).

Conferences

- Siddiqui, A.J., Boukerche, A. "Encoded Flow Features for Network Intrusion Detection in Internet of Things," IEEE 17th Annual Consumer Communications & Networking Conference, CCNC (2020).
- Siddiqui, A.J., Boukerche, A. "Adversarial Patches-based Attacks on Automated Vehicle Make and Model Recognition Systems," 16th ACM Symposium on QoS and Security for Wireless Mobile Networks (2020).

Relevant Prior Publications:

- Siddiqui, A. J., Mammeri, A., Boukerche, A. "Real-Time Vehicle Make and Model Recognition Based on a Bag of SURF Features," IEEE Transactions on Intelligent Transportation. Systems 17(11): 3205-3219, (2016).

Acknowledgements

All praises and thanks are due to the One and Only, Almighty Creator. And the best of salutations are due to the beloved Messenger of God, Prophet Muhammad (May peace and blessings of God be upon him) who inspires us to strive in spreading peace, love, kindness and generosity.

A number of sincere individuals have played a significant role in the accomplishments of this thesis. I express my deepest gratitude to my respected supervisor Prof. Azzedine Boukerche for his unending and selfless efforts in continuing to inspire, guide, and mentor me throughout my research. The invaluable feedback, constructive criticisms, and technical discussions with him at every step of my work have contributed a great deal to this thesis and nurtured me as a researcher. I sincerely thank Prof. Boukerche, NSERC-DIVA Strategic Research Network, NSERC-CREATE TRANSIT Network and the University of Ottawa for providing me financial support as a Research Assistant to pursue my research goals.

I owe a debt of gratitude to my beloved mother and father, Mrs. and Mr. Siddiqui, for being my source of inspiration always and for supporting me throughout my studies; and to my dear mother- and father-in-law, Mrs. and Mr. A. Ahmed, for the fruitful discussions in analyzing problems, advice and guidance. The constant help and guidance extended by my dear friends and colleagues including Dr. H. Abbasi, Dr. A. Mammeri (NRC), Dr. B. Pekilis (NRC), Y. Zhao, X. Ma, has helped me achieve various milestones throughout my PhD. In addition, I would also like to acknowledge the help and assistance from my colleagues and seniors at PARADISE Lab of UOttawa: Dr. F. Modesto, Dr. R. Coutinho, Dr. R. De Grande, C. Celes, Dr. S. Guan and others.

Finally, I thank my dear siblings, wife and kids for their prayers, encouragement and love that kept my morale high throughout the ups and downs during my research.

Dedication

To all those who are sincerely striving and working hard in the effort to spread peace, kindness, generosity and best of morals, in service to the creation of God, and to guide humanity to the eternal success.

Table of Contents

List of Tables

List of Figures

Glossary

A	
AE	Auto-Encoder
AI	Artificial Intelligence
ARP	Address Resolution Protocol
C	
CASS	Connected and Automated Surveillance Systems
CITD	Criteria-based In-Training De-activation
CNN	Convolutional Neural Network
CPAA	Cyber-Physical Adversarial Attacks
CPTD	Criteria-based Post-Training De-activation
D	
DDoS	Distributed Denial-of-Service
F	
FTP	File Transfer Protocol
I	
IDS	Intrusion Detection System
IoT	Internet of Things
M	
ML	Machine Learning
N	
NPS	Non-Printability Score

O

OS	Operating System

R

RITD	Random In-Training De-activation
RPTD	Random Post-Training De-activation

S

SIHFR	Symmetric Image-Half Flip and Replace
SSDP	Simple Service Discovery Protocol
SSH	Secure Shell Protocol
SSL	Secure Sockets Layer
SVM	Support Vector Machine

T

TCP	Transmission Control Protocol
TV	Total Variation

U

UDP	User Datagram Protocol

V

VMMR	Vehicle Make and Model Recognition

Chapter 1

Introduction

In order to ensure safety and security of life and property which is of paramount importance, internet-connected cameras are widely deployed for surveillance and monitoring in various environments such as roads or intersections (e.g., in smart cities and intelligent transportation systems), industrial facilities, residential or commercial properties, and critical infrastructures (e.g., railways, airports, malls, etc.). The surveillance cameras could be connected to each other, centralized control stations, cloud servers, mobile patrolling agents, etc., establishing an Internet of Things, or more specifically, an Internet of Surveillance Systems. The images or videos collected by the cameras are analysed by automated analytic systems that employ machine- or deep-learning-based algorithms. The recent success of deep convolutional neural networks (CNNs) in object detection and recognition has fuelled its use in automated surveillance systems such as Vehicle Make and Model Recognition (VMMR).

The connected surveillance systems are vulnerable to adversaries at both the physical world level and the cyber network level (as depicted in Figure 1.1). At the cyber network level, there are vulnerabilities to various network intrusion attacks (e.g., DDoS, Video Injection, Botnet, etc.) [146, 255]. At the physical world level, the surveillance systems are vulnerable to adversarial machine learning-based attacks (e.g., adversarially learnt patches that fool the machine learning algorithms behind the surveillance systems [96]) as well as camera physics-based attacks (e.g., blinding the camera through lasers [205]).

The works on network intrusion detection systems (IDSs) could be broadly categorized,

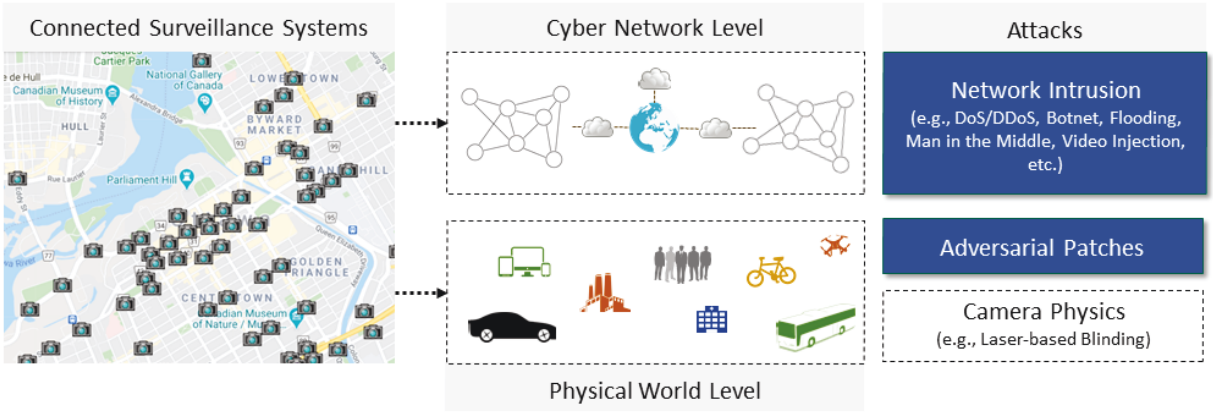


Figure 1.1: Conceptual overview of potential attacks against connected surveillance systems: at the cyber network level and at the physical world level. This thesis focuses on addressing the network level intrusions and the physical level adversarial patches-based attacks.

based on the point of deployment, as: (i) centralized, (ii) distributed, or (iii) hybrid [264]. In a centralized IDS, the detection modules or agents are hosted at the network edge devices or the border router (e.g., mobile edge computing servers deployed at eNodeBs, other base stations, or roadside units) [199]. On the other hand, in a distributed IDS, the detection modules or agents are hosted at the host (IoT) devices. A mix between the two is the hybrid strategy in which the detection agents or modules are distributed in a hierarchical fashion and hosted at the network edge as well as at the host (IoT) devices. The major advantages of a centralized IDS hosted at the network edge are the availability of richer computing, communication and storage resources, leveraging the advances in edge computing, as well as the ability to efficiently detect attacks originating externally (e.g., via the Internet) [199]. The solutions proposed to detect the network intrusions, based on machine learning techniques, could be grouped into: (i) supervised learning-based or (ii) unsupervised learning-based approaches. While the former learning approach leverages labelled datasets (and thus is suitable for known attacks detection), the latter does not rely on labelled data and uses the benign ground truth data instead (and hence is more suitable for detection of unknown attacks).

On the other hand, the works addressing the problem of adversarial machine learn-

ing attacks against deep learning-based surveillance systems such as Convolutional Neural Networks (CNNs)-based object detection or person detection could be classified into two groups: (i) input image pre-processing and (ii) adversarial training [190]. In the former approach, different methods have been proposed to pre-process input images to reduce or eliminate the influence of adversarial perturbations or patches, before feeding the input images to the detection models. In the latter approach, the object detection model is (re) trained with adversarial examples in order to make the detection model robust to those kinds of adversarially modified images. While the former approach adds some overhead (computational load and processing time) to a surveillance system, the latter approach generally requires re-training of the detection or classification models such as CNNs which may also be computationally expensive and time consuming. Thus, one of the main challenges in this area is to develop lightweight defense solutions that add minimal overheads and avoid re-training of the detection or classification models such as CNNs.

1.1 Thesis Statement

Research Problem: In the recent years, connected surveillance systems have drastically evolved along with the advancements in internet of things (IoT) and deep learning technologies. However, these developments have also lead to a rise in vulnerabilities to various kinds of attacks both at the cyber network level and at the physical world level. This poses danger not only to the devices but also to human life and property. One big challenge thus is to detect known and unknown network intrusion attacks as well as adversarial patches-based attacks through efficient yet accurate methods.

Key Idea: In this thesis, we propose the design of innovative solutions to detect known and unknown network intrusion attacks in internet of things such as connected video-based surveillance systems as well as to detect and mitigate adversarial patches-based attacks to defend such systems at the physical world level.

The methods to detect network intrusions attacks, whether supervised learning-based or unsupervised-learning based, rely on network flow features collected over certain time durations. While these raw flow features may be good at representing the statistical summaries of the network traffic over specific time windows, these do not explicitly represent

the temporally varying behavior of the devices in the network. Amongst the unsupervised learning-based methods, autoencoders (AEs) based on neural networks have recently gained attention. The AEs are leveraged to learn compressed latent representations of benign network flows, based on which malicious flows are detected. However, in the context of internet of things such as connected surveillance systems, the state-of-the-art AEs-based methods remain computationally expensive and time consuming. In the context of physical world level adversarial patches-based attacks targeting vision-based surveillance systems, the state-of-the-art defense methods face drawbacks due to computational and time-cost overheads. This may pose additional challenges for real-time surveillance systems, especially Vehicle Make and Model Recognition (VMMR).

Therefore, to address these limitations, we propose in this thesis innovative solutions that capture the temporally varying behavior of network devices, yield more discriminative feature representations, and that develop more efficient yet accurate ensembles of AEs. In addition, we study how adversarial patches-based attacks could impact the performance of a vision-based surveillance system trained for VMMR, as well as propose a lightweight solution to mitigate the effect of adversarial patches on VMMR systems.

1.2 Objectives

The main objectives of this thesis are to propose and develop efficient and accurate solutions to enhance the security of connected surveillance systems at both the cyber network level and physical world level. Specifically, we aim to design innovative network intrusion detection methods for known and unknown attacks in addition to a novel lightweight defense against adversarial patches-based attacks that target VMMR surveillance systems.

The design of methods to detect network intrusions through capturing the benign behaviors of IoT devices such as cameras is challenging due to time varying interaction patterns, e.g., surveillance cameras may be programmed to share the images more frequently at certain times of the day than at others. The success of AEs-based ensembles in problems of anomaly detection motivates their adoption for network intrusion detection. However, they remain to be computationally expensive for widespread deployment in internet of things such as video-based surveillance systems. The recent advancements in CNNs

have motivated their use in video-based surveillance systems such as person detection, face recognition, automated Vehicle Make and Model Recognition (VMMR), etc. While the vulnerability to adversarial patches-based attacks has been explored for surveillance systems such as person detection and face recognition, there has been no such study, to the best of our knowledge at the time of this thesis, in the context of VMMR.

In an overall sense, the work in this thesis aims to address these concerns and limitations towards developing more efficient and accurate defense solutions for an internet of things, especially connected surveillance systems, at both the cyber network and physical world levels. We intend to investigate the potentials of temporal codebook learning and informed ensemble pruning in dealing with the above-mentioned drawbacks in prior works on network intrusion detection. We intend to understand and identify the weaknesses in CNN-based surveillance systems, focusing on VMMR, with regards to adversarial patches-based physical world level attacks. Furthermore, we intend to design a lightweight defense method to mitigate the effect of such adversarial patches on VMMR.

1.3 Contributions

In light of the security issues in internet of things such as video-based surveillance systems (as discussed in the previous subsections), this thesis makes the following contributions, listed in the order they appear in the document:

Temporal Codebook-based Encoding of Flow Features: This contribution develops a novel flow feature representation, called the *TempoCode*, based on a temporal codebook which captures the key patterns in benign traffic over different time windows. The *TempoCode* transformation method measures the differences of flow samples from the key patterns in the learnt codebook. We develop a supervised machine learning-based ensemble of classifiers and optimize its parameters to learn to discriminate between benign *TempoCode* representations and different types of malicious ones. We study the effect of varying design parameters of *TempoCode* on classification scores and processing time, to suggest the optimal set of parameters. The proposed method is designed to serve in a centralized IDS, leveraging the compute and storage resources therein.

Adaptive Ensembles of Autoencoders: This contribution of the thesis concerns the development of methods that reduce the complexity of AE-ensembles while yielding high detection performance at low training, re-training and inference time costs. We propose two methods to select which AEs to de-activate in the ensemble: (i) *Criteria-based De-activations (CDA)* and (ii) *Random De-activations (RDA)*. In implementing these two methods, we develop two strategies of choosing when to de-activate the selected AEs: *Post-Training De-activation (PTD)* and *In-Training De-activation (ITD)*. We extensively evaluate the proposed methods and demonstrate their effectiveness, in comparison to state-of-the-art, in discriminating between anomalous malicious flows from the benign ones using two recent realistic datasets which were collected from real world deployments of IoT and connected surveillance cameras.

Evaluation of Adversarial Patches-based Attacks on VMMR: In this contribution, we introduce and investigate, for the first time to the best of our knowledge at the time of this thesis, an adversarial attack against CNN-based surveillance system responsible for VMMR, through adversarial patches that are learnt by adversarial machine learning. We demonstrate the effectiveness of the developed adversarial patches against VMMR through experimental evaluations on a real-world surveillance dataset. In addition, we study the effect of patch placement location on the attack's effectiveness. The developed adversarial patches achieve considerable reductions in VMMR recall scores. It is strongly believed that this work shall guide future studies in developing VMMR systems that are robust to adversarial learning-based attacks.

Lightweight Defense against Adversarial Patches-based Attacks on VMMR: This contribution involves the design and development of a lightweight defense method called *SIHFR* to eliminate the impact of adversarial patches on VMMR performance, leveraging the symmetry in vehicles' frontal (or rear) faces. Through experimental evaluations, we investigate the robustness of the proposed method under varying patch sizes and placement strategies, using a real-world surveillance dataset, and demonstrate its advantages over state-of-the-art.

1.4 Thesis Outline

The remainder of this thesis is organized as follows:

- Chapter 2 reviews state-of-the-art works on network intrusions and adversarial attacks in connected surveillance systems. It covers both supervised and unsupervised network intrusion detection methods. Furthermore, it introduces automated vehicle make and model recognition, a special application of surveillance systems. In addition, a review of adversarial patches-based attacks and defenses in object detection and image classification is provided.

- Chapter 3 presents and evaluates the proposed temporal codebook-based flow features encoding method, *TempoCode*, for supervised network intrusion detection. The effectiveness of *TempoCode*, in comparison to state-of-the-art, is established through extensive empirical evaluations using recent IDS datasets.

- Chapter 4 proposes and investigates novel methods to build adaptive ensembles of autoencoders for unsupervised network intrusion detection in IoT, focusing on connected surveillance cameras. The proposed methods achieve competitive results at much lower ensemble sizes, on real IoT datasets, thereby saving computational and time costs.

- Chapter 5 begins by providing an overview about the target surveillance system this part of the thesis focuses on, namely, Vehicle Make and Model Recognition (VMMR). This chapter investigates for the first time, to the best of our knowledge, the problem of adversarial patches-based attacks against VMMR systems. In addition, it proposes a lightweight defense method, *SIHFR*, designed to secure VMMR systems against such adversarial patches.

- Chapter 6 finally summarizes the contributions made by this thesis and presents some potential directions for future work.

Chapter 2

Network Intrusions and Adversarial Attacks against Connected and Automated Surveillance Systems

In the recent years, connected and automated surveillance systems (CASS) have been witnessing an unprecedented evolution owing to the advancements in internet of things and deep learning technologies. However, vulnerabilities to various kinds of attacks are also rising. This poses danger not only to the devices but also to human life and property. In this chapter, we provide a systematic review of the vulnerabilities in CASS based on categorizing the attacks into cyber-network and physical world levels. Adversarial attacks at the physical world level could be further grouped into cyber-physical (e.g., adversarial patches) and physical (e.g., laser-based camera blinding) ones. Based on the medium of surveillance, CASS could be classified as audio, visual or audio-visual surveillance systems. In this chapter, we provide a detailed discussion of the security issues in these surveillance systems, with a special focus on adversarial machine learning-based physical attacks. A taxonomy of cyber-physical adversarial attacks is proposed, based on which recent and representative works on the topic are reviewed and analyzed. Challenges and design guidelines for effective adversarial attacks are presented to motivate further research into developing robust defenses against such attacks. Moreover, this chapter provides a systematic review and discussion of the defense and mitigation methods proposed in the literature to secure CASS at the two broad levels. The paper concludes by presenting open issues and challenges in achieving resilient, robust and secure CASS. This survey shall equip researchers

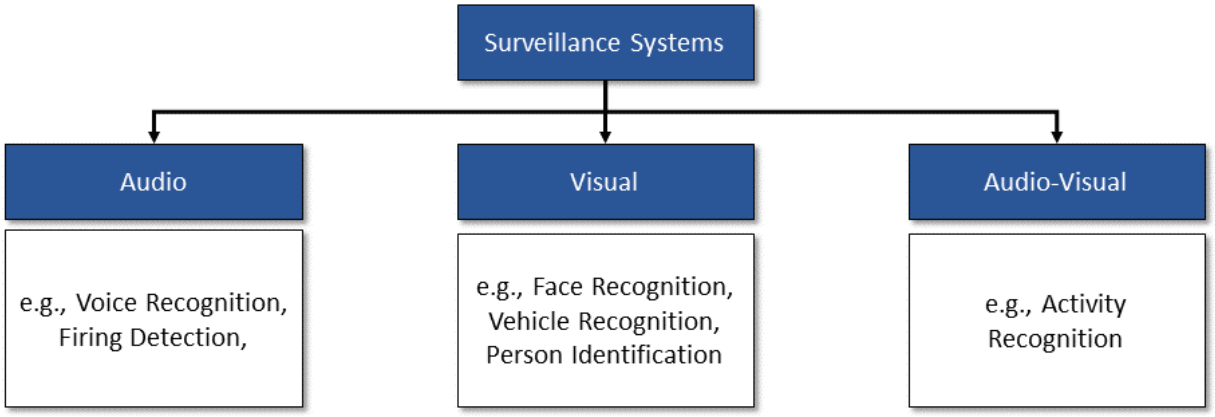


Figure 2.1: Three broad categories of connected and automated surveillance systems.

and practitioners in the area to analyze the security issues and defense measures in CASS, focusing on the three levels of vulnerabilities, and to inspire future works in enhancing state-of-the-art.

2.1 Background

The connected and automated surveillance systems may be deployed to serve various purposes, main ones being monitoring and target tracking. The monitoring operations are typically carried out for gathering information of activities happening at locations of interest. It could involve manual and automated techniques to detect suspicious objects, individuals, or anomalous, law-violating activities. The monitoring environments could include rooms, homes, offices, hospitals, malls, stadiums, airports, seaports, railways, borders, and critical infrastructure. The tracking operations are typically aimed at gathering forensic evidence and hunting for a target (e.g., a certain vehicle, individual or even a type of activity).

Although the monitoring and tracking operations have been traditionally carried out manually, recent developments in computer vision, deep learning and AI have led to automated systems that detect objects, activities and sound events that are of interest or are anomalous. Having such automated methods aids in making the surveillance operations

more efficient because human labor is reduced greatly, more footage can be simultaneously processed and response teams could be activated and mobilized more efficiently.

2.1.1 Connectivity and Topology

The connected and automated surveillance systems can be deployed in different ways. We describe here some of the typical ways of implementing the connectivity and topology. In general, the surveillance systems may be connected in three ways: (i) physically closed network, (ii) physically open network, and (iii) virtually closed network, In a physically closed network (PCN), the surveillance devices (e.g., cameras, microphones, or the digital recorders) do not bear a physical connection to the public Internet and are assigned private IP addresses. It does not allow a user to connect to the devices through the Internet. In a physically open network (PON), the surveillance devices bear connections to the Internet and are assigned public IP addresses, enabling access for a user via the Internet. In a virtually closed network (VCN), the surveillance devices are inter-connected through a private network, possessing private IP addresses, but allow access through the Internet using a virtual private network (VPN) [146]. The connectivity between the surveillance devices, digital recorders, routers, etc. could be through wired and/or wireless media.

The topology of the connected surveillance systems could be described based on their physical or geographical locations as: distributed or concentrated. In a distributed topology, the surveillance devices are scattered across a city, state, or even the globe. On the other hand, in a concentrated topology, the surveillance devices are physically located close to each other (e.g., within a home, mall or hospital).

The audio and/or video from the surveillance devices are transferred to the recorders or servers through analog or digital methods [146]. In the former method, a digital recorder gathers the analog signals of the audio/video footage and could bear connection to the Internet for remote access or visualization. In the latter method, the audio/video is transferred as packets, after processing and compression, using IP protocols.

2.1.2 Medium of Surveillance

Surveillance operations could be conducted over one or more of the following media: audio, visual, text, etc. In this thesis, we focus on visual (i.e., vision-based) surveillance. While vision-based surveillance provides richer information than audio, there are many events and incidents which may not be identifiable by video-based analysis alone. For example, a loitering vehicle may appear to be moving at normal speeds but its tires are skidding. The skidding may not be identifiable in the video alone but an accompanying audio stream could help in detecting the skidding sound.

2.1.3 Audio-based Surveillance

Audio-based automated surveillance finds its uses in detecting anomalies such as accidents and hazardous incidents. For example, the work of [122,123] developed a method to identify incidents of vehicle crashes and tire skidding based on audio stream analysis. Their method involves extracting discriminant features from the audio streams and representing them in a Bag of Words (BoW) style. The BoW-based representations are then used to detect anomalous events of interest.

Detection of abnormal events is important in railway environments as well. As such, the work of [173] presents deep learning-based methods to detect abnormal incidents such as screams, gunshots, glass breaks and sprays. Specifically, their methods leverage convolutional recurrent neural networks (CRNNs). Other applications or tasks of audio-based surveillance include urban sound analysis, event localization, speech emotion, and speaker identification/recognition.

2.1.4 Vision-based Surveillance

In vision-based surveillance, images or videos captured by various kinds of cameras and imaging devices are utilized for a plethora of tasks and applications such as image/video classification, object detection, activity recognition, anomalous action recognition, tracking of suspicious objects, trespassing detection, identity verification, etc.

The main components involved in a vision-based connected and automated surveillance system are: Data Acquisition, Pre-processing, Streaming, Transfer, Storage, Visualization, Analytics, Monitoring and Alarming. Depending on the scene under surveillance, different settings could be put in place, for example to transfer more packets per second upon detection of motion or certain activities. In this thesis, the focus is on a specific application of vision-based surveillance, i.e., automated vehicle make and mode recognition described further in Section 2.3.3.

2.1.5 Vulnerabilities of Connected and Automated Surveillance Systems

The connected and automated surveillance systems are vulnerable to adversaries at the cyber-network level and the physical world level (as depicted in Figure 1.1) At the cyber network level, there are vulnerabilities to various network intrusion attacks (e.g., DDoS, Video Injection, Botnet, etc.) [127, 130, 138, 146, 181, 217, 255]. At the physical world level, the surveillance systems are vulnerable to adversarial machine learning-based cyber-physical attacks (e.g., adversarially learnt patches that fool the machine learning algorithms behind the surveillance systems [96]). Additionally, camera physics-based attacks (e.g., blinding the camera through lasers [205]) and adversarial camera stickers [164] are some examples of vulnerabilities that could be exploited by adversaries .

2.2 Network Intrusions: Attacks and Defenses

2.2.1 Cyber-Network Intrusions

Adversaries may launch cyber-network intrusion attacks against connected and automated surveillance systems for various goals such as to conduct other kinds of attacks. Connected and automated surveillance systems can be vulnerable to various kinds of cyber-network attacks such as video injection, video replay, denial of service, botnet, and man in the middle. These attacks could be misused to compromise the visual feed, obstruct detection

of malicious events, activities or objects [106]. Table 4.2 presents some representative attacks on connected surveillance cameras from [180]

Video Injection

A live video feed of a connected and automated surveillance system could be compromised by an adversary that can inject recorded video clips into the live stream In such attacks, frames or clips of idle or benign events are pre-recorded and replayed to hide live events. Such attacks may go unnoticed even by human observers [188].

Denial of Service

There are various ways an adversary could launch denial of service attacks against a connected and automated surveillance systems. The camera devices itself could be compromised or disconnected from the network. A digital video recording server could be overloaded by spamming the server. Access to view, store or retrieve video feeds could be denied.

Botnet

Adversaries could infect connected cameras with botnet and malware to conduct large scale distributed denial of service (DDoS) attacks. For example, the Mirai botnet had exploited surveillance systems to launch serious DDoS attacks [151].

Man in the middle

These type of attacks involve intercepting the network traffic to conduct covert observations or launch attacks such as video injection and replay. Through man in the middle type of attacks, an adversary could manipulate, fake, or drop traffic in the network.

The solutions proposed to detect the network intrusions, based on machine learning techniques, could be grouped into: (i) supervised learning-based or (ii) unsupervised

Table 2.1: Description and Types of Attacks on Connected Cameras (Kitsune Datasets [180])

Attack Category	**Attack**	**Description**
Botnet Malware	Mirai	IoT device credentials exploited to inject Mirai malware and hunt for new victims
Reconnaissance	OS Scan	Hunts for vulnerabilities in hosts and operating systems in the network
	Fuzzing	Random commands used to hunts for vulnerabilities in cameras' web servers
	Video Injection	Other/recorded videos injected into the live video streams
Man in the middle	ARP MitM	ARP Poisoning exploited to intercept LAN traffic
	Active Wiretap	Exposed cables exploited to setup a covert network bridge (wiretap) to intercept LAN traffic
Denial of Service	SSDP Flood	Cameras exploited to generate UPnP advertisements to overload/spam the recording server(s)
	SYN DoS	A camera's web server is overloaded to disable the camera's video stream
	SSL Renego-tiation	A camera is overloaded with SSL renegotia-tion packets to disable its video stream

learning-based approaches. While the former learning approach leverages labelled datasets (and thus is suitable for known attacks detection), the latter does not rely on labelled data and uses the benign ground truth data instead (and hence is more suitable for detection of unknown attacks).

2.2.2 Defending against Cyber-Network Intrusion

The problem of intrusion detection has been a hot topic over the years, with rising attention due to the evolution of networks into an Internet of Things and the rising threats [27, 34, 64, 69, 73, 77, 161–163, 176, 177, 191, 224, 267]. The solutions proposed to detect the network intrusions, based on machine learning techniques, could be grouped into: (i) supervised learning-based or (ii) unsupervised learning-based approaches. While the former learning approach leverages labelled datasets (and thus is suitable for known attacks detection), the latter does not rely on labelled data and uses the benign ground truth data instead (and hence is more suitable for detection of unknown attacks).

Supervised Methods

In this section, we provide a brief overview of state-of-the-art methods in supervised IDS which are close in concept to one of our proposed method and highlight the uniqueness in our method.

A multi-layer semi-supervised framework for IDS is proposed in [267] based on pure cluster extraction, pattern discovery, fine-grained classification and model updating. They defined "pure" clusters as those in which a vast majority of their samples belong to the same class. They employed a hierarchical semi-supervised k-means algorithm to learn the pure clusters. The samples which did not fall into any pure cluster are then fed into the pattern discovery module in which a clustering-based method is used to find if the patterns are known (normal or intrusions) or unknown patterns. The fine-grained classification module then acts to classify the discovered unknown patterns. However, this step may require manual inspection to determine the class of the unknown patterns. The task of re-training any modules due to changes in traffic distribution is handled by the model updating module. Their experiments were based on the old KDDCUP99 dataset.

A lightweight and rule-based intrusion detection algorithm was proposed in [227] for networks of vehicles. In their method, vehicles aid in evaluation of behavior and reputation of their neighboring vehicles. An ensemble learning-based method is proposed by [185] who proposed statistical flow features and an AdaBoost ensemble comprising of Decision Tree, Naive Bayes, and Artificial Neural Network classifiers.

In [162], the authors proposed a two-stage intrusion detection system for Software-Defined IoT comprising of: (i) an enhanced swarm intelligence algorithm called the Improved Bat Algorithm for optimal features selection and (ii) a Random Forest classifier enhanced with a weighted voting mechanism. They used a downsampled version of KDD-CUP99 dataset. Software Defined Networking was also used for detection and mitigation of anomalies in the work of [133] which proposed a density peak-based clustering algorithm enhanced by sampling adaptation for detecting anomalies. In addition, the authors proposed an unsupervised clustering mechanism to select optimal features.

The authors of [197] proposed a clustered IDS based on Restricted Boltzmann Machine (RBC-IDS) as a deep learning-based solution to monitor critical infrastructures and detect intrusions in wireless sensor networks. Designed to work as a centralised IDS, RBC-IDS groups the sensors into a number of clusters and selects a cluster head in each cluster. The cluster heads are responsible of sending the sensor data (possibly after aggregation) to the central IDS. Their work is based on an old dataset which was not collected from an IoT environment. Moreover, the authors found the Restricted Boltzmann Machine to be slower than traditional machine learning-based methods. We compare (in Section 3.4.4) the performance of our method with that of a method which is related (yet more advanced) to [197] and uses deep auto-encoders on a realistic IoT dataset.

Another work addressing the problem of intrusions in a critical infrastructure monitoring application using wireless sensor networks is of [196] in which the authors investigated the development of an IDS based on a Reinforcement Learning (RL) algorithm called Q-Learning. However, their use of Q-learning as the RL algorithm presents some limitations. For example, the Q-tables could grow very large with a large number of states and/or actions. This would imply that with an increase in the number of states, the memory size required to save and update the Q-table would increase. Moreover, an extremely large amount of time would be required to explore each state for building the Q-table.

The work of [28] tackles the issues of intrusions in a connected vehicle cloud environment. It built an IDS comprising of a deep belief network for data reduction and an ID3-based decision tree classifier for classification. In Section 3.4.4 (Table 3.11) we show that the method proposed in this thesis outperforms DBN-based methods and an ID3-based method.

In a more general traffic classification context, the work of [276] introduced a Bag-of-flows model which groups together correlated flows. For classification, they aggregate the correlated predictions of Naive Bayes classifiers. Their work differs from ours in various aspects, most important one being that unlike our proposed method, they do not transform the flow features but simply discretize them in an objective to enhance classification accuracies.

In [27], the authors utilize feature quantization based on clusters in a Self-Organised Map (SOM) network. The SOM network is learnt first and then clusters of neurons are constructed through hierarchical agglomerative clustering. They assumed that the largest SOM cluster would represent benign traffic. So, if a test sample falls into this cluster, it is regarded as a benign one. However, unlike them, we don't consider the class of closest cluster to be the class of a sample (benign or malicious), but rather we take into account the similarities or distances to each cluster center (codeword), collecting as features a set of deviations of each training/testing flow from the cluster centers (codewords) that are temporally learnt.

Another work leveraging clustering of flows is that of [34]. However, their method is different from ours in the following aspects. Their clustering is based on destination IPv4 address prefixes, whereas we do not consider IP addresses in our method in order to tackle the attackers' ability to spoof and dynamically change IPs. Moreover, their feature transformation is based on approximating probability densities of individual flow-based statistical features. For this, they use the estimated probability density function of the respective feature in the closest cluster to which a sample flow belongs. In contrast to their approach, our feature transformation is based on capturing the flow features' distances from each of the different cluster centers.

Another clustering-based intrusion detection method called CANN was proposed by [165]. In their work, they transform a flow's features into a sum of distances of the flow

features from the cluster centers and from their k-nearest neighbors correspondingly, giving a one dimensional feature to be used by a kNN classifier. Although CANN results in a very low dimension feature space, its performance in terms of classification scores was limited compared to the performance of non-transformed flow features with other classifiers such as Support Vector Machines (SVM).

A recent study by [161] have tackled the problem of botnet identification through a clustering-based technique. In their method, they dynamically select subset of features expected to be more discriminative for the respective type of botnet. The clustering is performed on benign flows and malicious flows yielding the cluster centers as fingerprints of each application type (benign or botnet). They employ a similarity function that computes a distance-score of a sample flow from the cluster centers. Based on a closeness threshold, the class of the nearest cluster center is decided as a prediction of the sample's class. Their method is very similar to that of [278] who also employed a learnt codebook.

Given a sample flow's features, [278]'s method uses only the deviation from the closest cluster center and classifies it as anomalous if the deviation is greater than a threshold. However, this threshold has to be experimentally determined and may not be applicable to all types of attacks. Moreover, to evade detection, attackers could be mimicking benign flows by dynamically changing their flow features such that these may be within the threshold distance to atleast one of the benign clusters. In these cases, methods such as those of [161, 278] would suffer from decreased True Positive Rates. Hence, in our work, instead of dynamic feature subset selection, we transform the flow-based features into TempoCode representations which capture the deviation of each flow's features from benign fingerprints' or codewords' features and then leverage a machine learning-based algorithm to discriminate between the benign and malicious TempoCode representations.

Unsupervised Methods: Autoencoders for Intrusion Detection

The systems and methods in the literature concerning intrusion detection in IoT networks can be broadly classified into two approaches: (i) anomaly detection-based, and (ii) malicious traffic signatures-based. In the former approach, normal profiles of IoT devices are learnt using different methods, and a considerable deviation from these normal profiles

is suspected to be intrusions or malicious. In the latter approach, the system maintains a database of signatures or features of known botnets or malicious traffic and performs detections using these signatures. The former approach can deal with unknown botnets or attacks. However, there is a possibility of not detecting infected devices (e.g. in cases where the malicious traffic is imitating a device's normal traffic). The latter approach may be very accurate in detecting known attacks. However, its resiliency suffers in cases of new attacks whose signatures or features are not contained in the database.

The anomaly detection-based approaches gain preference due to the advantage of being able to detect unknown attacks. In this regard, neural networks-based autoencoders [137] have shown great success in various anomaly detection problems [223], e.g., in areas of medical image anomalies [262], outlier detection [279] as well as network intrusion detection [120, 274]

The work of NBa-IoT [175] proposed to build an autoencoder model for each device. In large-scale IoT networks, their approach may not be economically scalable. On the other hand, Kitsune [180] builds an ensemble of small autoencoders over different feature subspaces built based on feature correlations. A multi-layer and ensemble-based approach in [274] builds a stacked sparse autoencoder network for deep features extraction and a classifier based on a binary tree ensemble.

Allowing to learn benign patterns per device is essential to achieve tolerance to the heterogeneity of IoT devices. However, methods that enable scaling up or down adaptively are needed to ensure scalability and increased efficiency in resource-usage. The afore-mentioned works do not provide for mechanisms to adaptively scale up or down the ensemble of autoencoders which may be needed for efficiency purposes. To address these limitations with consideration of massive IoT devices, in Chapter 4, we propose four methods to realise an efficient ensemble which learns the best autoencoders (AEs) that can capture the patterns in normal behaviors of IoT devices of various types. These methods provide mechanisms to activate and de-activate the AEs in the ensemble in an unsupervised fashion in- or post-training.

2.3 Cyber-Physical Adversarial Attacks: An Emerging Challenge for Vehicular Surveillance Applications

Deep learning models designed for automated detection or recognition tasks in surveillance systems can be deceived by adversarially learnt perturbations that manipulate the model's classification decision. These adversarial perturbations can be introduced by an adversary in the digital and/or physical realm. These are thus referred to as Cyber-Physical Adversarial Attacks (CPAAs) in this thesis.

2.3.1 Taxonomy

This thesis proposes a taxonomy of cyber-physical attacks on connected surveillance systems, as shown in Fig. 2.2, based on the following dimensions and aspects: (i) targeted medium of surveillance, (ii) adversary's knowledge of target system/model (whitebox vs blackbox), (iii) spatial/temporal coverage of adversarial perturbation (non-localized/scattered or localized), (iv) adversarial objective (dodging vs impersonation, targeted vs non-targeted), (v) targeted task (classification, detection or others). The proposed taxonomy expands on the one introduced by [213]. Next, we provide a description of the mentioned categories.

Target Medium of Surveillance

Since surveillance could be based on different modalities such as visual, audio or hybrid, as described in Section 2.1.2, cyber-physical adversarial attacks on surveillance systems can be grouped based on the medium of surveillance targeted. The adversarial attacks could be targeted against visual or audio surveillance and hence grouped as *Vision-based CPAAs* or *Audio-based CPAAs*, respectively. As the names suggest, while adversarial attacks on the visual mode include perturbations that manipulate images, those against the audio mode include perturbations that manipulate the sounds being input to the surveillance system. The focus of this thesis is on vision-based systems and methods.

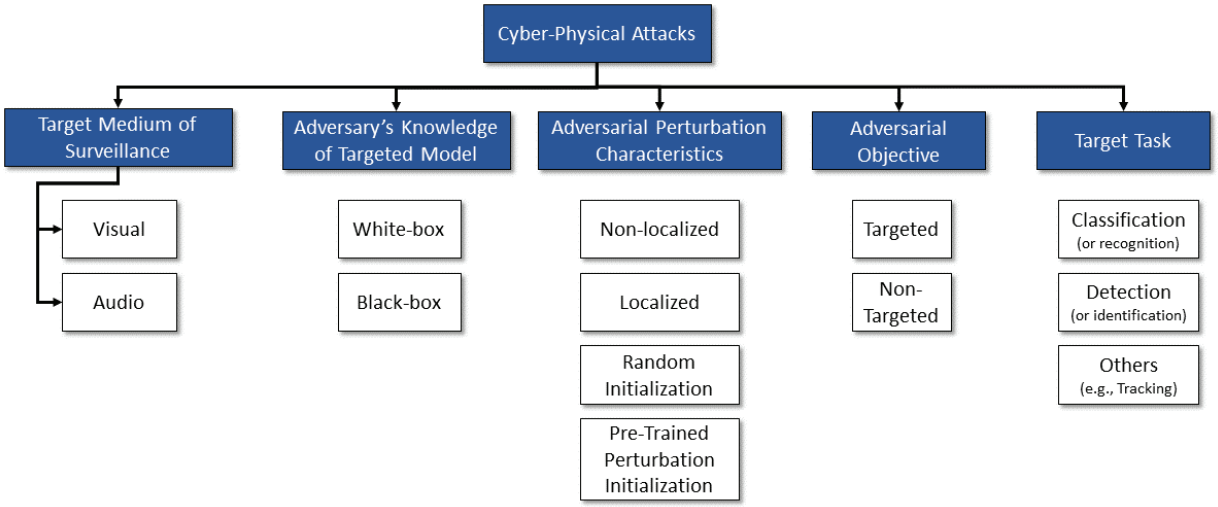


Figure 2.2: The proposed taxonomy of Cyber-Physical Adversarial Attacks

Adversary's Knowledge of Targeted Model

Depending on whether an adversary has the knowledge pertaining to the working details of a targeted model, adversarial attacks could be of two types: (i) white-box, or (ii) black-box.

White-box attacks: In these adversarial attacks, an adversary has access to the inner details of the deep learning model being used by a surveillance system. These include details about the model architecture, weights, gradients, loss functions used or training method.

Black-box attacks: In this type of attack, an adversary has no access to the targeted model and does not know the inner details of its architecture except for the model's output such as classification label or confidence score. Black-box attacks are typically conducted in two ways [213]: (a) query-based and (b) transfer-based. In the former, an adversary queries the target model a number of times to estimate the gradient [6]. In the latter, which is usually more difficult than the former, adversarial examples trained to attack a model in a white-box fashion are transferred to attack another model in a black-box fashion [200]. The success of such attacks may depend on the degree of similarity between the two models.

Adversarial Perturbation Characteristics

Adversarial attacks involve learning perturbations on input data samples such that the resulting samples are close to the original ones and that the adversarial perturbations are imperceptible to humans.

Adversarial perturbations' span/spread could either be localized or non-localized on the input. Localized perturbations manipulate a small portion of input data, e.g., a small patch on an image, On the other hand, non-localized perturbations manipulate scattered or entire portion of input data.

Adversarial attacks vary also based on perturbation characteristics such as initial patch configuration and perturbation scope. While some adversarial attacks may use perturbations pre-trained to attack other models to start with, some attacks may train these from scratch starting from randomly initialized ones. The scope of perturbation pertains to whether a separate perturbation is generated for each data sample, or a universal perturbation is learnt for the entire dataset of clean samples [271].

Adversarial Objective

Cyber-physical adversarial attacks could be designed to be targeted or non-targeted. In targeted attacks, the objective is to mislead the classifier to output a specific target class with high confidence or prediction score. Such attacks could be used for example to launch impersonation attacks where an adversary may trick the system to be mis-recognized as some specific target person.

In non-targeted attacks, the objective is to mislead the classifier to output any class other than the ground truth (i.e. actual) class. Such attacks cause the classifier to produce low confidence or prediction scores for the actual class while producing a higher one for any other class. Adversaries could use such attacks for dodging or evasion from the automated surveillance systems. For example, an adversary may want to evade being identified by a face identification system. In binary classification problems, targeted attacks and non-targeted attacks become equivalent.

Target Task

An automated surveillance system may have various tasks including classification, detection (or identification), and others such as tracking. In classification tasks, objective is to classify the data sample given, e.g., given an image (potentially containing several objects), to classify the most "salient" object such as a human, animal, or vehicle, etc. In detection, objective is to identify the objects or events of interest and localize them in the input. For example, in an image, to detect different types objects seen while locating them with bounding boxes on the image; in an audio, an example detection task could be to identify a particular sound event such as a gun shot or accident and marking the start and end times of the detected event in the data sample. Other tasks could include tracking of detected objects, events or activities over some time interval which may be useful for automated drones- or robot-based surveillance or target following. Adversarial attacks could be designed to target any of these tasks.

2.3.2 Challenges for Cyber-Physical Adversarial Attacks

To realize practical cyber-physical adversarial attacks, adversaries take into account various challenges pertaining to environmental conditions, sensor characteristics, spatial constraints, production errors, perturbation sharpness or smoothness and imperceptibility limits in physical world. The discussions on these challenges serve as design guidelines in investigating the development of (and defending against) adversarial patches or perturbations that are highly effective in real world.

When adversarial perturbations trained in the digital space are produced in real world, these challenges come into play and may influence attack effectiveness. In real world applications, an adversary may or may not have access to the sensors and/or the data pipeline. Adversaries that do not have access to the sensors and data pipeline are limited to developing and launching adversarial attacks in the real world. However, through cyber-network intrusions (discussed earlier in Section 2.2), an adversary could gain access to a sensor and its data pipeline to inject the perturbations digitally, e.g., through video injection attacks or man-in-the-middle type of attacks.

Environmental Conditions

The conditions in/around the scene where a target sensor is situated may introduce transformations to the adversarial perturbations or the perturbed samples which could render the perturbations to appear differently when captured by the sensor and fed into a deep learning model. Specifically, conditions such as lighting, objects' distance, their pose, sensor viewpoint, weather conditions such as snow, fog or rain add to the challenges of realizing effective adversarial perturbations in real world.

In the case of adversarial attacks on the visual mode of surveillance, natural light intensity, ambient light intensity and colors could influence how an adversarial perturbation or image may be captured by the target sensor. In some applications, objects may appear in the sensors' field of view at varying distances which would affect the size of an adversarial perturbation or patch in the image captured by the sensor. An adversarial perturbation/patch may appear to the sensor at varying viewpoints and perspectives. Moreover, there may be noise or blur introduced by the sensor capturing the input [254]. Hence, for effective adversarial attacks, these considerations need to be taken into account in training the adversarial perturbations.

Spatial constraints

Depending on what application is being targeted, the adversarial perturbation/patch placement space may be broad or limited. Adversarial patch placement space also depends on whether the attack places an adversarial patch/perturbation on the background regions or on the target objects.

For real world attacks, if an adversarial patch/perturbation is added on to the background, it may be possible that the target object or other objects may occlude that part of the background region thus rendering the attack ineffective. For attacks in digital space, an adversary may have the liberty to or could adapt the patch/perturbation placement location as needed. Thus, for real-world applicable adversarial attacks, some researchers argue that perturbations/patches are better to be added to target objects rather than to the background [213].

Production-related Errors

This challenge relates to the errors introduced while printing, producing or fabricating the theoretically (virtually) learnt adversarial perturbations or patches into the real world. For adversarial images, such errors may occur while printing the adversarial perturbations onto physical objects such as screens or cardboards.

Modern printing equipment have limitations with regards to the printable colors, i.e. their color gamut [1]. Moreover, some colors of the adversarial patch or perturbation may not be accurately produced in real world [213, 230]. To overcome this challenge, [230] introduced incorporation of printing related errors in the adversarial patch/perturbation generation process. Specifically, they proposed the use of non-printability score (NPS) in the objective function to be optimized.

The NPS measures how far an adversarial patch's pixel color values are from the given set of commonly printable colors [230]. Given an adversarial patch P, the NPS of its pixel $\hat{p}$ is formulated as:

$$NPS(\hat{p}) = \prod_{c \in C} |\hat{p} - c| \tag{2.1}$$

Here, c refers to a color from the set of commonly printable colors given by C. NPS will be high for a $\hat{p}$ that is far from any c and low otherwise. NPS has been used in road sign attacks [119], person detection attacks [254], steering decision attack [280], vehicle make and model recognition attacks [234].

For adversarial audio, errors or disturbances may occur while playing the adversarial sound perturbations from an audio source

Patch/Perturbation-related Sharpness or Smoothness

Generally, in most applications, clean images exhibit smooth and/or continuous changes in color [213] Patches or perturbations that exhibit low smoothness, i.e., have highly varying adjacent pixel values, may not appear as desired in the captured images. One reason could be the sampling noise due to which a camera may not be able to capture the extreme

[1] Color gamut refers to "the range of colors which a particular device can produce or record" [38]

differences in adjacent pixels. Hence, adversarial patches or perturbations that are not smooth may be ineffective when realized in real world. To overcome this challenge, [230] proposed using a smoothness ensuring function based on *Total Variation (TV)* such that variation of pixel values within adversarial patches or perturbations are kept low and smooth. This function is added to the objective function being optimized to generate the adversarial patch or perturbation. The TV of an adversarial patch or perturbation P is calculated as:

$$TV(P) = \sum_{p_{i,j} \in P} \left(\sqrt{(p_{i,j} - p_{i+1,j})^2 + (p_{i,j} - p_{i,j+1})^2} \right. \tag{2.2}$$

where $p_{i,j}$ denotes P's pixel value at the (i, j) coordinate Lower TV means higher smoothness and vice versa. A higher TV may lead to weaker concealment of the adversarial patch.

Imperceptibility Limits in Physical World

Some adversarial perturbations in the digital space are designed to be imperceptible to humans through small magnitude perturbations. However, ensuring that the target sensor is able to capture these small perturbations is a challenge.

For adversarial attacks based on patches or localized perturbations such as stickers or t-shirts, these could most likely be observable by humans. However, if adversaries design the perceptible perturbations or patches that can be printed and placed to appear as natural parts of the target objects, it may succeed in not arising suspicions by human observers. For example, an adversarial patch covering the entire hood of a vehicle, or an entire t-shirt, or a purse/backpack, or sunglasses/frames, etc.

Some researchers have started to attempt developing visually natural adversarial patches, e.g. the work of [149]. However, adversarial attacks with patches that are completely imperceptible in the physical world haven't been found in the literature so far to the best of our knowledge.

2.3.3 Automated Vehicle Make and Model Recognition

The objective of vehicle make and model recognition is to identify the make (e.g., Toyota) and model (e.g., Camry) of vehicle images captured by surveillance cameras. Although this problem may seem simple, there are several challenges particular to VMMR, as we shall describe below. In this section, we provide a brief review of VMMR works. The main modules in VMMR are the Features Extraction and Global Representation module, and the Classification module.

Security is a great concern in highly vulnerable areas such as parking lots of public spaces (e.g., malls, stadiums, airports, etc.). In these critical scenarios, in a smart [131], a CASS running over surveillance cameras' images can greatly assist the security personnel in identifying vehicles belonging to certain colors, types, makes, or models. Moreover, in cases where the police are searching for a target vehicle of a specific make or model, CASS would save considerable amount of time, resources and manpower. The mobile police vehicles equipped with video/images sensors could share the video/images [3,55,61,89,94,194,201] of target vehicles with other police vehicles and with roadside infrastructure, through various video dissemination techniques over connected vehicles [83, 124, 171, 218, 219, 236, 260]. In addition, for applications such as electronic toll collection, where different charges are applied to different types of vehicles, or vehicular traffic analysis/prediction [**?**, 11, 16, 246, 247, 249], vision-based CASS could serve as a complementary tool in improving efficiency of existing systems.

In an Internet of Vehicles (IoV), vehicles can share information amongst themselves and/or with roadside infrastructure [12, 105, 268–270] to provide critical services such as target tracking, monitoring or surveillance. The vehicle classification systems such as VMMR enable surveillant vehicles to recognize important information about targets. These targets can be even localized though various localization protocols [4, 22, 45, 50, 60, 79, 80, 86, 114, 115, 193, 225]. In order to meet the speed and fault-tolerance requirements of critical surveillance and monitoring applications, to enable reliable dissemination of information in a fast manner, protocols such as [9, 10, 30, 62, 63, 75, 81, 202] need to be utilized. In a dense surveillance scenario such as a large area with numerous surveillance cameras, or a large city with a number of surveillant vehicles, efficient synchronization between different nodes is required [42, 47]. To ensure connectivity of mobile surveillant

vehicles with each other and the infrastructure, ideas from mobility management studies such as [13–15,17–21,23–26,36,74,100,245,248,277] can be employed to reduce packet losses and latency. Since security of IoV-enabled surveillance systems is an essential requirement, owing to privacy concerns when it comes to sharing targets' information over the open wireless channels, trust-based security systems such as [43, 46, 52, 84, 85, 215, 216] could be used.

The problem of vision-based automated VMMR can be considered as a challenging multi-class image classification problem, in which a class is defined as a particular vehicle make and model. However, VMMR presents a more diverse and challenging set of issues than in other image classification problems. The various challenges in VMMR include *Ambiguity* and *Multiplicity*, as initially presented by [139]. Other issues include diversity of vehicle types, lighting or environmental conditions, occlusions (e.g., by pedestrians or other vehicles), or customized vehicle design or appearance. In the context of VMMR, multiplicity and ambiguity issues are described as follows.

- *Ambiguity*: With regards to VMMR, we classify the *ambiguity* problem into two kinds: (a) *Inter-Make Ambiguity*, and (b) *Intra-Make Ambiguity*. The former ambiguity refers to the issue of vehicles (models) of different companies (makes) having visually similar shape or appearance, i.e., two different make-model classes have similar front or rear views. For example, "Toyota Camry 2005" and "Nissan Cefiro 1999" are similar in visual appearance (see Figure 2.3). The latter kind of ambiguity results when different vehicles (models) of the same company (make) are similar in shape or appearance. For example, the "Altis" and "Camry" models of the "Toyota" make have similar front faces (see Figure 2.4). The similarity may not necessarily be in a visual sense only, but could also be in the feature space used to describe their images.

- *Multiplicity*: This problem occurs when a vehicle model (of the same make) has different shapes, designs or appearances. Figure 2.5 shows some examples of the multiplicity problem in NTOU-MMR Dataset [139].

In the recent years, researchers have initiated investigating deep learning and convolution neural networks (CNNs) to address the VMMR problem. Instead of using hand-designed features, they leverage CNN to learn (and represent) useful features. One such

(a) T Camry 2005 (b) N Cefiro 1999 (c) T Tercel 2005

(d) N Sentra 2003 (e) T Camry 2006 (f) F Mondeo 0

Figure 2.3: Inter-Make Ambiguity Problems between Toyota Camry 2005 and Nissan Cefiro 1999, Toyota Tercel 2005 and Nissan 2003, Toyota Camry 2006 and Ford Mondeo 0, in NTOU-MMR Dataset of [139]. "T", "N", and "F" stand for "Toyota", "Nissan", and "Ford", respectively.

(a) N Sentra 2003 (b) N Cefiro 1997 (c) T Altis 2008

(d) T Camry 2008 (e) T Camry 2006 (f) T Altis 2006

Figure 2.4: Intra-Make Ambiguity Problems between Nissan Sentra 2003 and Nissan Cefiro 1997, Toyota Altis 2008 and Toyota Camry 2008, Toyota Altis 2008 and Toyota Camry 2006, Toyota Camry 2006 and Toyota Altis 2006, in NTOU-MMR Dataset of [139]. "N" and "T" stand for "Nissan" and "Toyota", respectively.

(a) T Wish 2010 (b) T Wish 2009 (c) T Wish 2005

(d) T Camry 2008 (e) T Camry 2010

(f) F Mondeo 2 (g) F Mondeo 3

(h) H CRV 2003 (i) H CRV 2005 (j) H CRV 2009

Figure 2.5: Multiplicity Problems with Toyota Wish, Toyota Camry, Ford Mondeo, and Honda CRV in NTOU-MMR Dataset [139]. "T", "H", and "F" stand for "Toyota", "Honda", and "Ford", respectively.

work is of [239] who proposed some modifications to the inputs fed into the network for enhanced accuracies in comparison to traditional CNN. Initially, a 3D bounding box enclosing the vehicle is detected in the 2D camera image, representing three faces: front/rear, side, and roof. The image regions within the 3D bounding box faces are "unpacked" onto a 2D planar image. The unpacked image is fed into the network early on as input. Another input is the encoded viewpoint, derived from the 3D bounding box orientation. The viewpoint is encoded by three 2D vectors, each connecting the bounding box's center to the respective face's center. In addition, the viewpoint encoding is done through bounding box rasterization, where each face is rasterized in a separate color channel (R/G/B). The rasterized bounding box and viewpoint encodings are fed into the network after the convolution layers.

Their experiments with different combinations of above-mentioned additional inputs revealed different levels of accuracy improvements in comparison to a baseline CNN. Best results were achieved with the following modifications to the baseline CNN: (1) unpacked image as input, without feeding in the rasterized bounding box and viewpoint encodings; and (2) unpacked image as input along with the rasterized bounding box and viewpoint encodings. Other modifications included adding rasterized bounding boxes (without viewpoint encodings), or adding viewpoint encodings (without rasterized bounding box), with the original vehicle image or with the unpacked image as input. Besides VMMR, the authors also achieved fine-grained classification and verification of vehicles' sub-model and year.

In a fine-grained classification problem such as VMMR, alignment of same class objects and discrimination between different classes is greatly aided by learning objects' critical parts [152]. To this end, [152] propose an approach to learn a discriminative mixture of parts without depending on part annotations. A CNN is employed to extract features from the parts. Upon comparing two CNN architectures (CaffeNet [145] and VGGNet [237]), they observed VGGNet yielding best results on the Cars-196 dataset [153].

Another deep learning framework is proposed by [263]. It employs attributes-based [121, 159, 261] and multi-task learning [99] combined with information sharing. They proposed a data augmentation method to address the issue of data scarcity, to avoid overfitting problems in training deep CNNs. The fine-grained data is augmented by a large number of

hyper-class labelled auxiliary images. The "inherent attributes of fine-grained data" which are easy to annotate are taken as the hyper-class labels. One hyper-class label considered is the super-type of the classes (e.g., "cars"). The other hyper-class considered is related to the vehicle's pose (or viewpoint) in the image, referred to as "factor-type hyper-class", since pose is a hidden-factor of an image.

The work of [263] proposed a deep CNN model to address the high intra-class variance and low inter-class variance (ambiguity) issues. The key difference of their learning approach is the utilization of hyper-class augmented data and regularization between hyper-class classifiers and fine-grained classifiers. The authors obtained better accuracies with their Hyperclass Augmented CNN (HA-CNN) and Hyperclass Augmented Regularized CNN (HAR-CNN) models (which were trained from scratch on the relatively small-scale target data) in comparison to CNN-based works which employ networks pre-trained on other large-scale generic dataset. This indicates that features learned from generic large-scale datasets (such as ImageNet [222]) may not always be suitable for a specific fine-grained classification task such as VMMR. Moreover, even better accuracies were achieved when the HA-CNN and HAR-CNN models were not trained from scratch, but used a pre-trained model followed by fine-tuning on target data. Other works such as [152] built a classifier based on the max-margin template selection scheme of [101]. In [239], while the CNN model was directly used for classification, they employed a cosine distance-based method for vehicle "verification". Another work that uses CNN-based features is of [266]. The CNN-based features are used to train two classifiers: (i) Joint Bayesian, and (ii) SVM. The former was found to yield better results than the latter.

An important requirement for testing and evaluating VMMR approaches in real-world scenarios is to have a comprehensive dataset that represents real-world conditions (occlusions, varying weather and lighting), multiplicity and ambiguity issues, etc. In Table 2.2, we review and compare the different datasets used in representative VMMR works.

Table 2.2: Comparison of VMMR Datasets

Work	In/Outdoor	Weather		Lighting		Occlusions
		Rainy	Cloudy	Daylight	Dark	
[186]	-/Y	N/A	N/A	Y	N/A	N
[160]	-/Y	N/A	N/A	Y	N/A	N
[104]	Y/Y	N/A	N/A	Y	Y	Toll-gate Occlusion (Partial)
[272, 273]	-/Y	N/A	N/A	Y	N/A	Y (artificial occlusions)
[206]	-/Y	N/A	N/A	Y	N/A	N
[207]	Y/Y	N/A	N/A	Y	N/A	Partial
[203]	N/Y	N	Y	Y	N	N
[144] (Toy Cars)	Y/N	N	N	N	N	N
[2]	-/Y	N/A	N/A	Y	Y	N
[37]	-/Y	N/A	N/A	Y	N/A	N
[257]	N/Y	Y	Y	Y	Y	Y; Overcast Shadows
[275]	-/Y	N/A	N/A	Y	Y	Y
[226]	-/Y	N/A	N/A	Y	N	N
[139]	Y/Y	Y	Y	Y	Y	Y; Overcast Shadows
[167]	-/Y	N/A	Y	Y	N/A	N/A
[168]	-/Y	N	Y	Y	N	N
[102]	Y/Y	Y	Y	Y	Y	Y; Overcast Shadows
[134]	N/Y	N/A	Y	Y	Y	N
CompCars [266]	Y/Y	N/A	Y	Y	Y	N/A
[39]	-/Y	N/A	N/A	N	Y	N/A
BoxCars [239]	-/Y	N/A	Y	Y	N/A	N/A

Although CNN-based approaches yield promising results in VMMR, these are highly vulnerable to adversarial attacks such as those based on adversarial patches which we introduce and discuss in Section 5.2.

2.3.4 Convolutional Neural Networks for Object Detection and Image Classification Models

In this section, we describe the architecture and learning methodology of a CNN-based popular object detection model called YOLO [209] and a popular image classification model known as ResNet50 [135]. Although there have been many CNN models proposed in the recent years, some of which are variants of the afore-mentioned ones, we describe the working of these two representative models to provide a background to the adversarial patches-based attack which is presented and discussed in Section 5.2.

YOLO

A popular object detector known as YOLO [210] represents the group of CNNs designed to work as single shot detectors. In such end-to-end detectors, a single pass of the image (through the layers of the model) is used to simultaneously predict the existence of objects of interest (i.e., objectness score), location of objects of interest (i.e., bounding boxes enclosing the objects) as well as class scores (i.e., probabilities of the object belonging to a certain class). The pioneering model of [210], followed by its fully convolutional variants YOLOv2 [211] and YOLOv3 [212] have demonstrated great success in object detection problems. Here, we describe the architecture and working of YOLOv2.

Given an input image, the output of YOLOv2 is a grid of much smaller resolution (32 $\times$ smaller) than that of the input image, The grid cells represent the predictions over the corresponding parts of the input image. Each grid cell produces five output vectors, one for each "anchor point". The anchor points are rectangles of different aspect ratios to accommodate objects of various aspect ratios. Each of the five output vectors contains predicted values for the following: position of the bounding box center (relative to the anchor point) represented by x_{offset} and y_{offset}, dimensions of this bounding box (width w

and height h), objectness score p_{obj} and class probabilities p_{cls1} to p_{clsN} (for N classes). The prediction scores are optimized using a cross entropy loss function and backpropagation.

ResNet50

The ResNet convolutional neural network is amongst the earlier deep neural networks that were designed to overcome the vanishing gradient problem faced with increasing network depths. The ResNet architecture enabled training deep networks beyond the 5 layers of AlexNet [154], beyond the 19 layer VGG [238] and 22 layer GoogleNet (InceptionV1) [250].

The striking feature of the ResNet architecture is the introduction of residual learning by the use of residual blocks [136] that employ shortcut connection between layers (skipping three layers) to perform identity mappings. As the name suggests, there are 50 layers in ResNet50, where the final layer is composed of a fully connected layer with number of nodes equal to the number of classes considered.

2.3.5 Adversarial Patches-based Attacks on Object Detection and Image Classification Models

Very recently, the problem of adversarial attacks against deep learning models has gained attention. Although several studies have been made recently on adversarial attacks against image classifiers, general object detectors, road signs, face recognition systems, to the best of our knowledge, our work is first to investigate adversarial attacks against VMMR systems using adversarially learnt patches. In this section, we present some of the recent works on adversarial patches-based attacks against object detection and image classification CNNs.

As discussed in Section 2.3.1, adversarial perturbations could either be spatially *non-localized* or *localized*. Moreover, adversarial patches are used to perform two broad types of attacks, namely: *targeted* and *non-targeted*. The *targeted* adversarial attacks could be used to conduct impersonation attacks, i.e., pretending or appearing as the target object or class. The *non-targeted* adversarial attacks on the other hand could be used to cause dodging attacks, i.e., the adversary just intends to avoid being detected or classified as its true self [230].

Non-Localized Adversarial Perturbations

The initial works on adversarial attacks were focused on non-localized adversarial perturbations to attack deep learning models trained for image classification. Table 2.3 provides a summary of the representative works on non-localized adversarial perturbations.

Table 2.3: Summary of representative works on Non-localized Adversarial Perturbations

Work	Objective	Perturbation Characteristics	Task	Real world
[251] (L-BFGS)	Non-targeted, white-box	non-lozalized	image classification	x
[126] (FGSM)	non-targeted	whitebox; non-localized	image classification	x
[220]	Non-targeted;	whitebox;	image classification	x
[157]	targeted	whitebox	image classification	x
[156] (BIM)	non-targeted	non-localized; whitebox; blackbox potential;	image classification	✓
[156] (ILLCM)	whitebox; blackbox potential; targeted	non-localized	image classification	✓

The generated perturbations are kept small to be imperceptible and the resulting perturbed images are usually termed as adversarial examples [251]. Given a clean input sample $I \in \mathbb{R}^m$, a target deep learning model f, an adversarial perturbation $\eta = \hat{I} - I$ is generally generated according to the following box-constrained optimization problem:

$$\min_{\hat{I}} ||\hat{I} - I|| \tag{2.3}$$

$$\text{subject to } f(\hat{I}) = \hat{l} \tag{2.4}$$

$$f(I) = l \tag{2.5}$$

$$l \neq \hat{l} \tag{2.6}$$

$$\hat{I} \in [0, 1]^m \tag{2.7}$$

where the model's output labels for I and $\hat{I}$ are represented by l and $\hat{l}$, respectively. The distance between the original and perturbed sample is denoted by $||.||$. The goal of this optimization problem is to cause misclassification while minimizing the adversarial perturbations. Different variants of this optimization problem have been proposed in the literature to generate adversarial perturbations.

The work of [251] introduced the vulnerability of deep neural networks to adversarially perturbed examples. In their work, adversarial perturbations were learnt based on the following optimization problem:

$$\min_{\hat{I}} c||\eta|| + L_f(\hat{I}, \hat{l}) \tag{2.8}$$

$$\text{subject to } \hat{I} \in [0, 1]^m \tag{2.9}$$

where L_f denotes the model f's continuous loss function. Their work used the L-BFGS method to solve the above box-constrained problem. Based on line-search, minimum $c > 0$ was approximated such that $f(I + \eta) = \hat{l}$ is satisfied by the minimizer η in the above problem. However, the attack method of [251] had limitations in terms of computational and time expense, mainly due to the linear search method that was employed.

[126] proposed adversarial examples generation based on the fast gradient sign method (FGSM). The underlying target model's gradients are used by the gradient sign method. To each pixel of an image I, small perturbations are added or subtracted depending on the pixel's gradient sign (i.e., positive or negative). By the addition of perturbations to

an image in the gradient's direction, model classification can be mislead. Equation 2.10 expresses their proposed perturbation η:

$$\eta = \hat{I} - I = \epsilon \cdot sign(\nabla_I L(\theta, I, l)) \tag{2.10}$$

where $\nabla_I L$ denotes the gradient of the loss function of the target model with respect to the original input I, l is the ground truth label of I, θ represents the parameters of the target model, ϵ denotes the perturbation magnitude, $\hat{I} = I + \eta$ is then the generated adversarial sample. Backpropagation can be used to compute the perturbation.

A positive gradient sign indicates that the model's loss will increase with increase in the pixel's intensity . Similarly, a negative gradient sign indicates increase in the model's loss with decrease in the pixel's intensity. Models that are vulnerable to the gradient sign method are those that favor linearity, such that the relationship between class prediction score and a pixel intensity of the input is treated linearly. Neural networks architectures such as LSTMs, maxout networks, or those with ReLU activation units are some examples. Adversarial examples learnt against a target model have been shown to be transferable to attack other models that were trained on the same task though having different architectures [182].

[126] had provided a suggestion to make models robust to such attacks which involved including adversarial examples in the models' training datasets. Researchers have proposed variants of the FGSM method, e.g., [220] developed the fast gradient value method and [157] modified FGSM to conduct targeted attacks.

While the works discussed above focussed on adversarial perturbations in the digital space to data being fed into the classifiers, the work of [156] studied adversarial attacks on practical real world systems such as those taking input from cameras and other sensors. One of the method proposed by [156] is called the Basic Iterative Method (BIM) which extends [126]'s FGSM method. In BIM, the FGSM is applied a number of times (i.e., over a number of iterations) using small step size and clipping of pixel values. To keep the intermediate results of each step within an α neighborhood of the original image, these are clipped as shown in the equation:

$$\hat{I}_0 = I \tag{2.11}$$

$$\hat{I}_{N+1} = Clip_{I,\alpha}\{\hat{I}_N + \epsilon \cdot sign(\nabla_I L(\hat{I}_N, l))\} \tag{2.12}$$

where $\epsilon = 1$ was used in their work to limit the pixel value change by 1 in each step, $Clip_{I,\alpha}\{\hat{I}\} = min\{255, I + \alpha, max\{0, I - \epsilon, \hat{I}\}\}$ does the clipping in each iteration to limit the adversarial image change and to keep the result within L_∞ α-neighborhood of original input I [156]. The number of iterations was determined as the $min(\alpha + 4, 1.25\alpha)$. It was heuristically chosen and reported to be enough so that the edge of the α max-norm ball was reached by the adversarial example. Moreover, it was sufficiently restricted for ensuring manageable computational costs of their experiments.

The BIM method of [156] yielded non-targeted attacks. In contrast, their Iterative Least-Likely Class Method (ILLCM) was proposed to learn targeted attacks. For an image I, the least-likely class l_{LL} is chosen as:

$$l_{LL} = \arg\min_l p(l|I) \tag{2.13}$$

where $p(l|I)$ denotes the model's prediction score on I and l_{LL} for a well-trained classifier model would usually be very different from the ground truth label. For neural networks with cross-entropy loss, the iterative procedure is modified to:

$$\hat{I}_0 = I \tag{2.14}$$

$$\hat{I}_{N+1} = Clip_{I,\alpha}\{\hat{I}_N - \epsilon \cdot sign(\nabla_I L(\hat{I}_N, l_{LL}))\} \tag{2.15}$$

The real world experiments of [156] involved printing the adversarial samples and capturing these through a smartphone camera. The attack evaluations were made both under white-box and black-box settings. The authors found that their iterative methods were less robust to photo transformations than the FGSM method.

Table 2.4: Summary of representative Localized Adversarial Perturbations

Work	Objective	Perturbation Characteristics	Task	Real world	Targeted models
[244]	non-targeted and targeted	black-box; localized (one-pixel)	image classification	×	
[96]	targeted	white-box training/testing, transferability testing in blackbox fashion; localized (patches); universal (regardless of background)	object detection	✓	White-box: InceptionV3, ResNet50, Xception, VGG16, VGG19; Black-box: whitebox training on 4, blackbox test on 5th YOLOv2 Dataset: ImageNet
[254]	non-targeted; evasion	white-box; localized (patches); transferable; random initialization	person detection	✓	Faster RCNN, YOLO trained on MS COCO dataset
[166] (DPATCH)	targeted and non-targeted	whitebox training/testing,\\location-independent; transferability testing in blackbox fashion; randomly initialized	object detection	×	Faster-RCNN [214], YOLO [210] trained on Pascal VOC 2007 dataset [**?**]

Continued on next page

Table 2.4 – continued from previous page

Work	Objective	Perturbation Characteristics	Task	Real world	Targeted models
[230]	targeted and non-targeted; impersonation and dodging	whitebox and blackbox training/testing; transferable; localized (on eyeglasses)	face recognition/ID	✓	Deep neural networks, Face++, Viola-Jones Face Detector
[240]	targeted (false detections) and\\non-targeted (disappearance);	transferable; localized whitebox training/testing; blackbox testing	road sign recognition (stop signs)	✓	Faster-RCNN, YOLO, YOLOv2 [209]
[32]	targeted;	localized; 3D-printed	object detector	✓	InceptionV3 [**?**]

Localized Adversarial Perturbations

In non-localized adversarial perturbations, usually all pixels are perturbed while keeping the overall change restricted. However, localized adversarial perturbations are located in narrow regions or patches of input, or even on very few pixels. Table 2.4 provides a summary of representative works on localized adversarial perturbations.

With a goal to achieve imperceptibility, [244] used differential evolution (DE) and introduced the One-Pixel Attack. The optimization problem it seeks to solve is

$$\max_{\eta^*} f_{\hat{l}}(I + \eta) \tag{2.16}$$

$$\text{subject to } ||\eta||_0 \leq d \tag{2.17}$$

where $f_{\hat{l}}(I)$ denotes the classifier's prediction probability for class $\hat{l}$ given a sample I, and d is a small number that restricts the number of pixels perturbed ($d = 1$ in the one-pixel attack). The goal of this problem is to find the pixel to perturb and the associated modification strength. Since DE does not require gradient information from the target model, nor does it require the target model's loss function to be known, the one-pixel attack of [244] can be classified as a black-box attack.

The pioneering work of [96] developed targeted adversarial attacks by learning patches (i.e., a group of pixels) that cause the classifier to produce a specific target class as output. These patches were applied in real physical scenes with great effectiveness. These patches completely replaced an image part. The patch training process involves optimizing (maximizing) the target class' expected probability. Specifically, the authors used a variant of [33]'s Expectation over Transformation (EOT) framework to train an adversarial patch $\hat{P}$. The objective function to optimize is given as

$$\hat{P} = \arg\max_{P} \mathbb{E}_{I \sim \mathbf{I}, T \sim \mathbf{T}, O \sim \mathbf{O}}[log Pr(\hat{l}|A(P, I, O, T)] \tag{2.18}$$

where $\mathbf{I}$ is the set of training images, $\mathbf{T}$ represents a distribution over patch transformations, $\mathbf{O}$ denotes a distribution over image locations, and $\hat{l}$ is the target class. $A(\cdot)$ is the patch applicator that takes as input: an adversarial patch, an image, image location to place the patch on, and any transformations to apply on the patch (e.g., scaling or rotations); and produced the resulting image after applying the transformed patch at the specified

location. The expectation is over images, image locations and patch transformations, and thus results in a universal perturbation that works regardless of the background. The adversarial patches were trained in a white-box fashion but were tested for transferability to attack models not seen in training. The adversarial patch developed was printed and tested in real world in which it demonstrated lower effectiveness than in the digital space, possibly due to the fact that printability was not considered in optimizing the patch. The models targeted include InceptionV3 [**?**], ResNet50 [135], Xception [**?**], VGG16 [**?**], and VGG19 [238] trained on ImageNet dataset [222].

The work of [254] studied adversarial patches targeted against automated surveillance cameras, specifically person detection. It was demonstrated that adversarially learnt patches could fool a popular object detector such as YOLOv2 [209] causing it to miss detecting persons in the input images. The goal of the optimization problem involved in learning the adversarial patch includes maximizing printability, minimizing total variation within the patch, and minimizing the detector's objectness or classification score. Since this method is based on backpropagation and requires knowledge of the model architecture, it can be regarded as a white-box attack.

The authors of [254] investigated three approaches to make the object detector not detect persons: (i) person class' classification probability was minimized, (ii) objectness score was minimized, and (iii) in which both (i) and (ii) were minimzed together. The targeted model was YOLOv2 object detector that was trained on MS COCO dataset . The authors observed that with approach-(i), the resulting adversarial patch resembled a particular class (other than the person class) which may not be transferable to attack other models that may not be trained for that class. The patch learn with approaches (ii) and (iii) seemed to be more generic, i.e., not resembling a particular class. Their results indicated that varying patch placement positions in the training process of the patches is important for effectiveness in real world.

Building upon the work of [96], adversarial patches that could perform both targeted and non-targeted attacks were proposed in DPATCH [166]. To learn patches for non-targeted adversarial attacks, DPATCH seeks to maximize the object detector's loss with respect to the ground truth class labels and bounding box parameters. The optimization problem to train a non-targeted adversarial patch $\hat{P}_u$ and a targeted adversarial patch $\hat{P}_t$

is given by following equations respectively:

$$\hat{P}_u = \arg\max_P \mathbb{E}_{I\sim\mathbf{I},T\sim\mathbf{T}}[L(A(I,T,P);l,B] \tag{2.19}$$

$$\hat{P}_t = \arg\min_P \mathbb{E}_{I\sim\mathbf{I},T\sim\mathbf{T}}[L(A(I,T,P);\hat{l}_t,\hat{B}_t] \tag{2.20}$$

where $A(\cdot)$ is the applicator similar to the one used in Equation 2.18, $\mathbf{T}$ is a distribution of transformations (specifically, [166] consider shifting), l and $\hat{l}$ denote respectively the ground truth and targeted class labels, B and $\hat{B}_t$ represent the ground truth bounding box and targeted bounding box parameters, respectively. For $\hat{P}_u$, the targeted model's loss to the ground truth class is maximized. For $\hat{P}_t$, the targeted model's loss to the target class $\hat{l}_t$ and target bounding box parameters $\hat{B}_t$ is minimized. In their work, the authors demonstrated how DPATCH trained to attack one CNN model (e.g., Faster RCNN [214]) could effectively attack another model (e.g. YOLO).

In [230], the authors proposed learning adversarial regions shaped as eye-glasses frames to attack face recognition models. Face recognition and identification is extensively used in surveillance applications. The authors' goal was to produce adversarial attacks that have negligible imperceptibility and that can be applied in real world. The adversarial regions are placed on the face as printed eye-glasses and are trained to conduct targeted and non-targeted attacks. With targeted adversarial printed eye-glasses frames, an adversary could perform an impersonation attack. On the other hand, with non-targeted adversarial printed eye-glasses frames, an adversary could perform a dodging attack (i.e., to be mis-identified as any other face).

For targeted (i.e., impersonation) and non-targeted (i.e., dodging) attacks, the optimization problems used by [230] are given by the Equations 2.21 and 2.22 respectively:

$$\arg\min_\eta \sum_{I\in\mathbf{I}} \Big(L_{softmax}(f(I+P),\hat{l}_t)) + \alpha\cdot TV(P) + \beta\cdot NPS(P) \Big) \tag{2.21}$$

$$\arg\min_\eta \sum_{I\in\mathbf{I}} \Big(-L_{softmax}(f(I+P),l_I)) + \alpha\cdot TV(P) + \beta\cdot NPS(P) \Big) \tag{2.22}$$

where $\hat{l}_t$ denotes a target class, l_I is the ground truth class, P is the adversarial perturbation, $NPS(P)$ is defined as in Equation 2.1, $TV(P)$ is defined as in Equation 2.2, and $L_{softmax}$

is softmax loss score defined as:

$$L_{softmax}(f(I), l_I) = -log \quad \frac{e^{\langle h_{l_I}, f(I)\rangle}}{\sum_{l=1}^{N} e^{\langle h_l, f(I)\rangle}}\Bigg)\Bigg(\tag{2.23}$$

where $\langle a, b\rangle$ represents inner product of vectors a and b, h_l denotes class l's one-hot encoded vector, and N is the number of classes. Gradient Descent algorithm [41] was used to solve the above optimization problems which essentially is an iterative algorithm that involves gradient evaluation and solution updates in the steepest descent direction. It is stopped either by restricting the number of iterations or upon convergence. Through experimental evaluations, the developed adversarial perturbations were shown to reduce the performance of both face recognition and face detection models.

Adversarial posters and stickers were proposed by [240] to cause object detectors to not detect stop signs - a potentially lethal attack against connected and autonomous vehicles. Moreover, their work trained adversarial stickers that were placed on flat objects (not stop signs) and caused object detectors to mis-detect these as stop signs.

The objective function of the optimization problem used in [240] to learn the adversarial posters and stickers can be generally represented as

$$\arg\min_{P} \lambda TV(M_I \cdot P) + NPS(M_I \cdot P) + \mathbb{E}_{I\sim\mathbf{I}} L(f_\theta(I + T_I(M_I \cdot P)), \hat{l}_t) \tag{2.24}$$

where M_I is a mask that spatially restricts the perturbation δ to the target object's surface, NPS and TV refer to non-printability score (Equation 2.1) and total variation (Equation 2.2) respectively, and the last term is the average value of the loss function $L(\cdot, \cdot)$ over all images in $\mathbf{I}$, T_I is a transformation function, $f_\theta(\cdot)$ is the target model's output, y^* is the targeted class. The loss function $L(\cdot, \cdot)$ is slightly modified depending on whether the goal is disappearance of a stop-sign or false-appearance of one. The authors reported loss of perturbation details and hence reduced attack effectiveness in real world due to environmental conditions such as those discussed in Section 2.3.2,

Unlike the previously discussed adversarial attacks which are based on 2D adversarial images, [32] developed 3D-printed adversarial objects to deceive object detectors. Specifically, a turtle-like 3D-printed adversarial object was trained to fool the objector to misclassify it has a rifle even from various angles. Trained using their proposed Expectation

over Transformation (EOT) algorithm, the adversarial examples were optimized across varying transformations such as rotations, scaling, etc. Such attacks could be used to "hide" suspicious, illegal or harmful objects from an automated surveillance system and hence demand attention towards developing robust defense mechanisms.

The adversarial patches-based attacks against VMMR systems could be launched by physically placing the printed patches on the vehicles or by digitally placing the patches on the captured images. The digital placement of patches could be achieved by compromising the surveillance camera network, e.g., through video injection attacks or man-in-the-middle type of attacks. Recent works have studied the problem of network intrusions in IoT and camera networks (e.g., [29, 146, 155, 158, 163, 198, 224, 233, 253]). The focus of Chapter 5 is on defending against attacks launched by placing the adversarial patches on the vehicles regardless of whether it is done physically or digitally. Nonetheless, in training the adversarial patches, the patch transformation and update module (as described in Section 5.2) factors into account real-world considerations of physical patch placement and appearance.

The works mentioned above study adversarial attacks against deep neural networks that are purposed for object detection, image classification or focused on specific applications such as face recognition, person detection, stop sign detection, etc. The problem of VMMR is different from these application domains and poses a peculiar set of challenges such as multiplicity and inter-class or intra-class ambiguities [92, 235], as discussed in Section 2.3.3.

On the other hand, in the case of stop signs for example, these commonly have the same shape and color. In the case of person detection, though the appearance of "person" objects varies (e.g. due to size, clothes or skin), there is a general outline or figure of persons. In the case of face recognition, multiplicity issues may occur due to factors like aging or facial hair, etc. Moreover, there is a wider area of placing adversarial patches on vehicles in comparison to other objects such as stop signs or persons.

Hence, to learn adversarial attacks against deep learning models that have been trained to overcome the VMMR challenges becomes more complicated. To the best of our knowledge, no prior study has investigated adversarial patches-based attacks against VMMR systems. We believe this thesis shall pave the way forward for future studies in developing adversarially robust VMMR and surveillance systems.

2.3.6 Defense methods against adversarial patches-based attacks

In the literature, only a few defenses could be found, to date, against physical world adversarial patches-based attacks that target CNNs. We review state-of-the-art defenses against such patch-based adversarial attacks and qualitatively discuss the limitations of these works in comparison to our method.

The work of [1] recently proposed a defense method that involves extracting ally patches from input images through counter-processing them based on their intrinsic information contents. In their method, from each input image, a set of patches are extracted to feed the object detector. These set of patches may include patches which enclose the adversarial patch fully or partially. While an adversarial patch partially enclosed in an ally patch is most likely to be ineffective in tricking the detector, a fully enclosed one may lead to a wrong output from the detector for that particular ally patch. However, since the final classification output is based on the predictions from each patch in the alliance, the effect of the adversarial patch may most likely be eliminated. A major limitation in this method is that the object detector network has to be executed multiple times per input image. Moreover, this method would require the object detector or classifier network to be trained to detect or classify objects of interest based on their partial views.

In Local Gradient Smoothing (LGS) [190], the authors leverage the observation that within the adversarial patches, the image gradients are usually large due to sharp changes in pixel values within the patches. Gradient smoothing is applied to the image regions exhibiting such a behavior. Their method outperformed other defense methods such as Digital Watermarking, JPEG Compression, Total Variance Minimization, and Feature Squeezing.

Table 2.5: Summary of Representative Works on Defending against Cyber-Physical Adversarial Attacks

Work	**Description**	**Possible Overhead**	**Adversarial Patches**
[1]	Object detector has to be executed on multiple patches per input image; May require the classifier to be re-trained to classify objects based on partial views.	High	✓
[132]	Regions with large classification loss gradients are masked out; In absence of adversarial patches, objects of interest may be wrongly masked out, possibly causing loss of information.	Low	✓
[113]	Uses JPEG compression to overcome adversarial pixel perturbations; Different regions of an image are compressed using random compression levels at test time; Not effective against adversarial patches-based attacks.	Low - Medium	×
[129]	Before an image is fed to a classifier, image transformations such as bit-depth reduction, JPEG compression, total variance minimization, and image quilting are applied; Ineffective against localized large variations such as those caused by adversarial patches.	Low - Medium	×
[234] (Our Proposed Method)	Eliminates adversarial patches by using the 'cleaner' symmetric image half; Reduces information loss that occurs if regions are simply masked out; Avoids classifier modification or re-training; Lightweight, adding minimal processing time overhead. (see Chapter 5)	Low	✓

The work of [132] observed that near the pixels perturbed by adversarial patches, gradients of classification loss with respect to the input image tends to exhibit large values. This behavior was leveraged along with saliency maps to estimate regions where an adversarial patch may be located. The authors of [132] developed a digital watermarking (DW)-based method to detect such regions with large gradients and mask out these regions from the image. However, in cases of no patch attacks, the saliency map would point towards an object of interest, the processing (and masking) of which may cause reduction in detection performance in clean images. A good defense method is expected not only to reduce the effect of adversarial patches, but also to achieve an accuracy on clean images as close to that achieved without the defense method on clean images.

Some works such as [113] have studied the use of JPEG compression which uses Discrete Cosine Transform (DCT) to eliminate high-frequency components. Although it was shown to defend against the effect of adversarial image perturbations, such methods are not effective against adversarially learnt patches-based attacks [190]. Similarly, works such as [128] which employed Total Variance Minimization, JPEG compression and image quilting, were also found to be ineffective in cases of localized large variations as caused by adversarial patches.

The problem with these approaches such as [132] is that they mask out the image regions, causing information loss. If the models are not trained to work with such missing pieces of information, then the clean accuracy also suffers. Contrary to these approaches, we propose a defense method that effectively replaces the suspected attacked regions of an image using its cleaner symmetric half, leveraging the symmetry in vehicles' frontal (or rear) faces. Table 2.5 provides a summary of the limitations in related works and highlights the contributions of this thesis.

Chapter 3

TempoCode: Temporal Codebook-based Encoding of Flow Features for Network Intrusion Detection

In the recent years, the Internet of Things has been becoming a vulnerable target of intrusion attacks. As the academia and industry move towards bringing the Internet of Things (IoT) to every sector of our lives, much attention needs to be given to develop advanced Intrusion Detection Systems (IDS) to detect such attacks. In this chapter, we propose a novel network-based intrusion detection method which learns patterns of benign flows in a temporal codebook. Based on the temporally learnt codebook, we propose a feature representation method to transform the raw flow-based statistical features into more discriminative representations, called *TempoCode*. We develop an ensemble of machine learning-based classifiers optimized to discriminate the malicious flows from the benign ones, based on the proposed TempoCode. The effectiveness of the proposed method is empirically evaluated on a state-of-the-art realistic intrusion detection dataset as well as on a real botnet-infected IoT dataset, achieving high accuracies and low false positive rates across a variety of intrusion attacks. Moreover, the proposed method outperforms several state-of-the-art works based on the used datasets, proving the effectiveness of Tempo-Code over raw flow features, both in terms of accuracies and processing speeds.

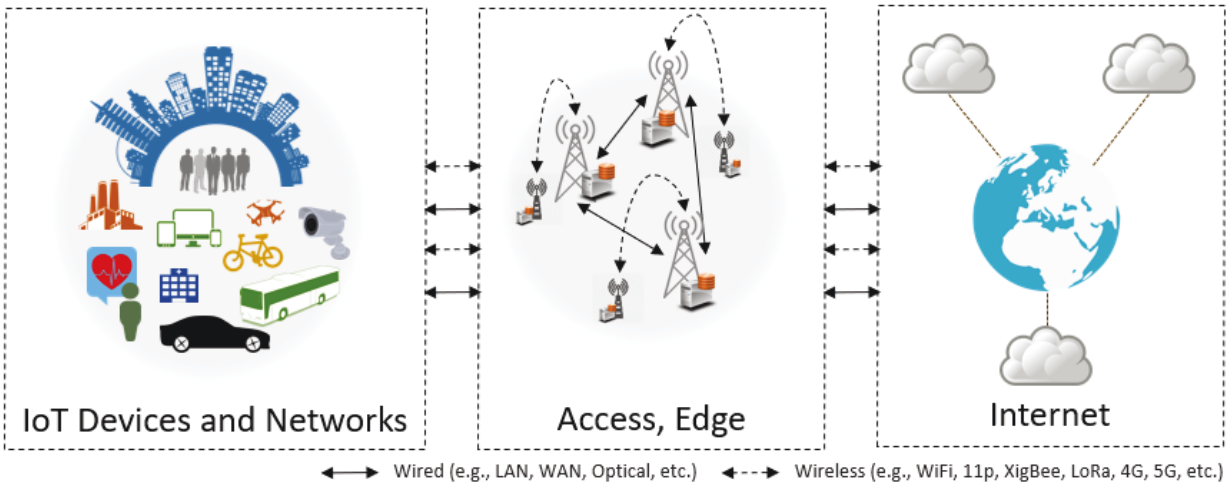


Figure 3.1: An architectural overview of the evolving Internet of Things landscape. The diverse range of IoT devices and networks (e.g., surveillance cameras and systems) connect to the access points (such as base stations, eNodeBs, RSUs, etc., equipped with edge computing resources) which can communicate amongst themselves and to the larger Internet.

3.1 Introduction and Background

The unprecedented evolution of networks with a growing plethora of connected devices and things are reshaping the landscape of an Internet of Things (IoT). Ranging from devices such as indoor or outdoor surveillance cameras, electrical and mechanical appliances, mobile user-worn devices such as smart watches or health monitors, to connected vehicles and vehicular components, industrial systems, and connected smart cities, the IoT landscape is continuously evolving (see Figure 3.1).

Due to the increasing diversity of devices, networks and services in an IoT ecosystem, the vulnerabilities of each constituent technology could be agglomerated, giving rise to novel threats and attack vectors [72,143,192,232]. This poses danger not only to the devices but also to life and property. Consider these recent reports for example. A large pool of internet-connected devices were compromised to conduct distributed denial of service (DDoS) attacks on critical networks [178]. Another serious example is of the Mirai botnet-based attack which exploited IoT devices to attack many popular web-based services and

platforms that became inaccessible [147].

The targets of such DDoS attacks could include critical infrastructure, banks, healthcare institutions, smart cities and internet of connected vehicles and things [28,73,169,217]. For example, connected vehicles have been shown to be vulnerable to being controlled by a remote malicious attacker who could shut down a moving car or lock/unlock doors [195]. In another worrisome example, smart (and connected) toys were found to have security and privacy flaws that could be exploited by an adversary to maliciously control the toys or cause serious privacy infringements [231].

In light of these incidents, researchers in academia and industry have devoted a great deal of resources to develop novel solutions for intrusion detection in IoT, to secure IoT from different types of intrusions [71, 217, 241, 258]. The various IDSs could be broadly classified in terms of their placement strategy as: centralized, distributed, or hybrid. In the centralized IDS placement strategy, the detection modules or agents are hosted at the network edge devices or the border router (e.g., mobile edge computing servers deployed at eNodeBs, other base stations, or roadside units). On the other hand, in the distributed strategy, the detection modules or agents are hosted at the IoT devices. A mix between the two is the hybrid strategy in which the detection agents or modules are distributed in a hierarchical fashion and hosted at the network edge as well as at the IoT devices. The main advantages of a centralized IDS hosted at the network edge are the availability of richer computing, communication and storage resources, leveraging the advances in edge computing, as well as the ability to efficiently detect attacks originating externally (e.g., via the Internet).

In order to address the problem of intrusion attacks in IoT, this chapter proposes a novel method to detect intrusions by transforming flow-based features into more discriminative representations and designs an ensemble of classifiers based on these to differentiate between benign and malicious flows. The proposed method is designed to serve in a centralized IDS, leveraging the compute and storage resources therein. The main contributions of this chapter are summarised as follows:

- We propose a novel flow feature representation, called the *TempoCode* based on unsupervised learning of a temporal codebook which captures the key patterns in

benign traffic over different time windows. The TempoCode transformation method measures the differences of flow samples from the key patterns in the learnt codebook.

- We study the effect of varying design parameters of TempoCode on classification scores and processing time, to suggest the optimal set of parameters.
- We develop a machine learning-based ensemble of classifiers and optimize its parameters to learn to discriminate between benign TempoCode representations and different types of malicious ones.
- We demonstrate the effectiveness of the proposed method in distinguishing benign flows from different types of intrusion attacks through empirical evaluations on two realistic, real-world and recent datasets (CICIDS2017 [229] and NBaIoT [175]).

The remainder of the chapter is organised as follows. In Section 2.2, we provide a brief discussion of recent related works, followed by presenting the proposed method in Section 3.2. In Section 3.3, we describe the experimental setup, details and specifications of the datasets used, different attack scenarios considered in this work, impact of design parameters, and the performance metrics to be used for evaluation. After presenting the results and discussions in Section 3.4, the chapter finally makes concluding remarks and outlines future work directions in Section 3.5.

3.2 Proposed Method

In a quest to overcome some of the limitations in prior works discussed above, the focus of this chapter is on designing a novel method to transform flow-based features into more discriminative representations. In this regard, this chapter proposes the *TempoCode*, a temporal codebook-based method of encoding flow features which captures the key patterns of benign flow features in an unsupervised, temporally learnt discriminative codebook. The proposed method builds upon enhancing the Bag-of-Features (BoF) model [111] in the context of network flow features-based IDS (BoF is alternatively referred to as Vector Quantization (VQ) in some works). The BoF model has been proven effective in other domains such as image classification, object recognition [189], etc.

Based on the temporally learnt codebook, the flow features are transformed into TempoCode representations which are then used to train an ensemble of SVM classifiers. Since the next generation IoT will embrace a heterogeneity of devices which will use a diversity of protocols and standards [217], our goal in this work is to build upon flow features that are not dependent on a specific protocol. Moreover, devices in IoT may exhibit time-varying behavior in terms of the traffic flows they generate. For example, at certain times of the day, the sensors or devices may be exchanging data at a higher rate than at other times. Similarly, in certain situations such as upon detection of a specific event, cameras or acoustic sensors may be triggered to exchange data at a higher rate. Considering such characteristics of IoT flows, we design a temporally learnt codebook which captures the key patterns in benign traffic over different time windows. The following subsections describe the steps in more detail. An overview of the proposed method's pipeline is depicted in Figure 3.2.

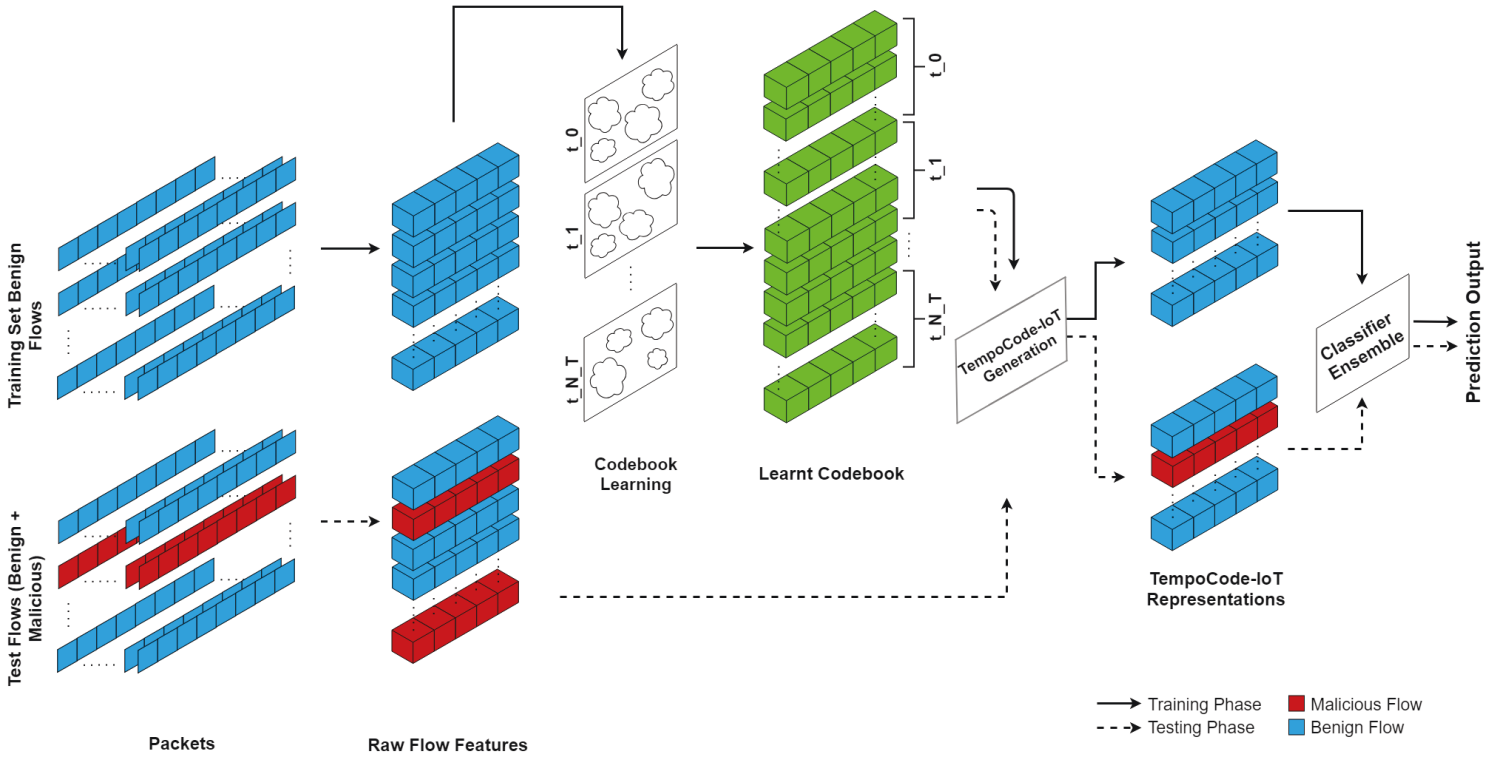


Figure 3.2: An overview of the TempoCode-based intrusion detection pipeline

The main steps for intrusion detection based on the proposed method are: (1) Flow-based features extraction, (2) Temporal Codebook Learning, (3) TempoCode Generation, and (4) Classifier Ensemble Training/Testing. The subsections below shall elaborate on these steps.

3.2.1 Flow-based Features Extraction

As depicted in Figure 3.2, the first step is to extract the flow-based features from the packets. To evaluate the proposed TempoCode feature transformation method, in this work, we use different sets of features as collected in two recent datasets: CICIDS2017 [229] and NBaIoT [175, 180]. These features include Flow Duration, Number of Packets in forward direction, Flow Inter-arrival time, as well as their statistical summaries such as mean and standard deviation (stdev) over certain intervals of time.

To allow for a fair comparison of our method with related works based on the selected datasets, our experimental evaluations on each dataset utilize the flow features contained in the respective dataset. The selected flow features have been popularly used in intrusion detection works. Using these as raw features, we demonstrate the benefits of the proposed TempoCode transformed feature representations over the raw features

The motivation to choose the mentioned datasets with different sets of features stems from the following reasons: (i) recent related works on intrusion detection have utilized these datasets, (ii) to establish the effectiveness of the proposed TempoCode feature transformation method with different sets of features, (iii) to demonstrate the effectiveness of proposed TempoCode method for IoT as well as other networking environments. Below, we provide an overview of the features in both datasets, while we give a deeper description of these datasets in Section 3.3.1.

Features in CICIDS2017 Datasets: The following flow-based features of CICIDS2017 datasets were selected and extracted (over every time interval of 1 second) using the CICFlowMeter tool [125], based on the findings of [229]:

- Flow Duration
- Packet Length (min, mean, stdev in forward and backward direction)

- Subflow Bytes (in forward direction)
- Flow Inter-Arrival Time (min, mean, stdev in forward and backward direction)
- Active_min, Active_mean
- Init-Win-Bytes (forward and backward),
- Flag Counts (ACK, PSH, SYN)
- Number of packets and bytes per second (forward and backward direction)

Features in NBaIoT Datasets: The NBaIoT data-sets [175, 180] contain similar features based on packets and flows (further details can be found in [175]). The below set of features are extracted over different damped time-window sizes, yielding a set of 115 features in total:

- Packet Size (mean and variance; outbound direction)
- Packet Count
- Packet Jitter (mean, variance and count of packet inter-arrival times)
- Packet Size (magnitude, radius, covariance, correlation coefficient; inbound and outbound

From each j-th record in a dataset, the set of features F_j, is represented as in Equation 3.1 below (where $n = N_F$ represents number of features). For example, f_{1j} could represent flow duration of j-th flow sample over a time window.

$$F_j = \{f_{1j}, f_{2j}..., f_{nj}\} \tag{3.1}$$

Table 3.1: TempoCode: Symbols and Notations

Symbol	Description
F_j	Set of features for j-th sample in dataset
N_F	Total number of features, i.e. dimensionality of F_j
f_{nj}	The n-th feature if F_j
t_{dur}	Size of time window for temporal codebook learning
N_T	Total number of time-windows
t_i	Time window i, where $i = 0, ..., N_{T-1}$
N_{ct}	Number of benign key patterns to learn from t_i
$\mathbf{C_T}$	The temporal codebook
$cw_{t_i,k}$	k-th key pattern (codeword) in $\mathbf{C_T}$ corresponding to time window t_i
$CSize$	Temporal codebook size = $N_T \cdot N_{ct}$
TC_j	TempoCode representation of F_j
$q_{t_i,k}$	Distance between $cw_{t_i,k}$ and F_j

3.2.2 Temporal Codebook Learning

The proposed method of transforming flow features into TempoCode representations first involves capturing key patterns of benign flows over different time windows in a temporal codebook which is learnt in an unsupervised manner. Below, we first describe the temporal codebook learning procedure, followed by the TempoCode transformation method.

The benign flow features are grouped according to successive (non-overlapping) time windows determined by t_{dur} (e.g., if $t_{dur} = 1$, its an hourly window). The total number of time windows is given by N_T which would depend on the training dataset and t_{dur}. For example, in the case of hourly windows, $t_{dur} = 1.0$, and a training dataset of 10 hours, then $N_T = 10/1.0 = 10$. The time windows are denoted by indices t_i (where $i = 0, ..., N_{T-1}$).

In each time window t_i, a clustering method such as K-Means is applied to learn N_{ct} key patterns as codewords to represent the benign traffic in t_i. In the K-Means clustering method, the initial set of cluster centres is chosen randomly. Nearest neighbours for each cluster centre are found from the data points. In cases where a data point is found to

be close to more than one cluster centre, it gets assigned to the closest one only. In each cluster, an average of its member data points is calculated and set as a new cluster centre. With the new cluster centres, each data point is re-grouped to the (new) centre closest to it. This process of updating cluster centres followed by re-grouping of data points based on the updated centres is repeated a number of times to ensure stability of the clusters. The cluster centres (from each time window) are stored as codewords $cw_{t_i,k}$. In this way, we obtain the temporal codebook $\mathbf{C_T}$, as expressed below, where $\mathbf{C_T} \in \mathbb{R}^{N_F}$

$$\mathbf{C_T} = \{cw_{t_i,k} | i = 0, ..., N_T; k = 1, ..., N_{ct}\} \tag{3.2}$$

The temporal codebook size $CSize$ can then be given by $N_T \cdot N_{ct}$, considering that we use a fixed N_{ct} in all time windows. The procedure to learn $\mathbf{C_T}$ is given in Algorithm 3.1.

3.2.3 TempoCode Generation: Temporal Codebook-based Encoding of Flow Features

Leveraging the temporal codebook $\mathbf{C_T}$ learnt above, the flow features are transformed into TempoCode representations based on their distances from the learnt benign patterns across different time windows. The procedure to generate TempoCode representations for training and testing dataset samples is shown in Algorithm 3.2. The raw features of a flow, F_j are compared to all the codewords of $\mathbf{C_T}$, where the deviations are captured to form the TempoCode representation, summarised in the equation below:

$$TC_j = \{q_{t_i,k} | i = 0, ..., N_T; k = 1, ..., N_{ct}\} \tag{3.3}$$

where a bin ($q_{t_i,k}$) holds the distance of F_j from $cw_{t_i,k}$. In this manner, a TempoCode representation is a set of distances of a raw feature vector to each of the codewords in $\mathbf{C_T}$, organised in a time-ordered fashion. The TempoCode representation is computed as follows, where $dist(.,.)$ is a distance metric such as euclidean distance:

$$q_{t_i,k} = dist(cw_{t_i,k}, F_j) \tag{3.4}$$

Algorithm 3.1 Temporal Codebook Learning

Input: $\mathbf{F} = \{F_j | j = 0, ..., N_s\}$, $CSize$ (Size of $\mathbf{C_T}$), t_{dur} (Time window), N_s is the total # of training samples, N_{s_i} (Number of samples in time window i)

Output: $\mathbf{C_T}$

Initialisation :

$\mathbf{C_T} = \{\}$

Grouping $\mathbf{F}$ *by time-window* t_{dur}

$\mathbf{F_T} = \{\}$

for $i = 0$ to N_T **do**

    $\mathbf{F_i} = \{\}$

    for each F_j in time window t_i, $j = 0$ to N_{s_i} **do**

        $\mathbf{F_i} = \mathbf{F_i} \cup F_j$

    end for

    $\mathbf{F_T} = \mathbf{F_T}.append(i, \mathbf{F_i})$

end for

Learning TempoCode Codebook

$\mathbf{C_t} = \{\}$

for each $\mathbf{F_i}$ in $\mathbf{F_T}$, $i = 0$ to N_T **do**

    $\mathbf{C_t} = Cluster(\mathbf{F_i}, N_{ct})$

    $\mathbf{C_T} = \mathbf{C_T}.append(i, \mathbf{C_t})$

end for

return $\mathbf{C_T}$

3.2.4 Classifier Ensemble Training

The TempoCode representations obtained from previous steps are then used to train an ensemble of machine learning-based classifiers (see Figure 3.2) such as Support Vector Machines (SVM) [97,256] using the *scikit-learn* machine learning library [204]. The codebook, and the respectively generated TempoCode representations are expected to be discriminative enough in the feature space to aid the classifier in learning the differences.

Algorithm 3.2 TempoCode Generation

Input: $\mathbf{F} = \{F_j | j = 0, ..., N_s\}$ (Training or testing samples), N_F (Dimensionality of F_j), $\mathbf{C_T}$,N_{ct} , N_T

Output: $\mathbf{F_T}$ (the TempoCode representations)

Initialisation :

$\mathbf{F_T} = \{\}$

TempoCode Transformation :

for each flow sample F_j in $\mathbf{F}$ **do**

  $F_{T_j} = []$ ▷ the TempoCode representation of F_j

  for each time window t_i, $i = 0, ..., (N_T - 1)$ **do**

    for each $cw_{t_i,k}$ in $\mathbf{C_T}$, $k = 0, ..., (N_{ct} - 1)$ **do**

      $F_{T_j}[k + i \cdot N_{ct}] = dist(cw_{t_i,k}, F_j)$

    end for

    $\mathbf{F_T} = \mathbf{F_T}.append(F_{T_j})$

  end for

end for

return $\mathbf{F_T}$

The SVM is originally a binary classifier that has proved its effectiveness in many classification problems. It learns the support vectors by leveraging kernel functions such as Radial Basis Functions (RBF). The support vectors are basically a subset of the training Tempo-Codesamples that represent the best separation between two classes. A test TempoCode sample is classified based on its distance from these support vectors. A single multi-class SVM classifier is built by collecting many such binary classifiers, depending on the number of classes in the dataset. We determine the optimal parameters for SVM

through extensive cross-validation experiments, owing to the imbalanced nature of the dataset used.

In this work, we designed an ensemble of multi-class SVM classifiers, a method based on Bagging [95]. Each classifier in this ensemble is trained on a random subset of the training dataset. The prediction of each constituent classifier is then combined through voting to produce the overall classification output.

We choose SVMs as the base classifiers motivated by their proven generalization ability, robustness, and success in achieving globally optimal solutions [35]. As such, they have found popular use in diverse applications such as multimedia analysis, object detection and recognition, medical image analysis, etc.

The motivation to adopt ensemble learning stems from the following reasons: (i) Training an ensemble of multiple classifiers on smaller subsets of training data could be done in parallel and faster, (ii) An ensemble has better generalization ability when compared to that of the individual learners (classifiers) [281].

In the testing phase, the TempoCode representations for test samples (which include both benign and malicious ones) are generated using the codebook $\mathbf{C_T}$ learnt above and passed on to the ensemble of classifiers. Then, each of the classifiers adds a vote to its predicted class. The class with the highest votes is assigned as the predicted class (benign, malicious, or a specific attack class) of the test flow sample.

3.2.5 Computational Complexity

The complexity of the proposed methods can be analysed by looking at the complexity of the main steps involved. The Temporal Codebook Learning step is only carried out in training phase. In this work, the clustering process involved in codebook learning employs the K-Means method which has an average complexity of $O(CSize \cdot n \cdot N_{iter} \cdot N_F)$ and worst case complexity of $O(n^{CSize+2/N_F})$ [31], where n is the number of (training) samples, $CSize$ is the codebook size (i.e., total number of cluster centres) and N_{iter} is the number of iterations of the K-Means method.

The TempoCode Generation step involves using the learnt temporal codebook to transform raw features into their TempoCode representations mainly by computing the devia-

tions of raw features from the codewords of the codebook. If the complexity of computing the distance of a N_F-dimensional sample from $CSize$ codewords (each of dimensionality N_F) of the codebook $\mathbf{C_T}$ is given by $O(CSize)$, then the complexity of TempoCode generation step for n (training or testing) samples would be $O(n \cdot CSize)$.

As for the complexity of Classifier Ensemble Training and Testing step, we look at the complexity of SVM classifiers which we employ as base classifiers in this work. The computational complexity of training SVM depends mainly upon the number of support vectors. It involves a quadratic term and a cubic term because the asymptotic number of support vectors (n_{sv}) grows linearly with the number of samples (n). When the parameter C is small, n_{sv} grows atleast like n^2 and when C becomes large, it grows atleast like n^3 [40]. In testing (prediction), the complexity of executing each SVM classifier is around $O(n_{sv} \cdot CSize)$ (each support vector has dimensionality $= CSize$).

Thus, the overall complexity of training by the proposed method is non-linear whereas testing (i.e., execution) is linear with respect to the number of (training or testing) samples n.

3.3 Experimental Setup

The diversity of devices in IoT adds to the challenges of developing security solutions. The mix of low-, moderate- and high-traffic generating devices, possibly running on different operating systems and protocols, and the fact that these devices could change their behaviors of exchanging data depending on various conditions, makes it difficult to discriminate between malicious flows from the benign ones.

Moreover, to the best of our knowledge, there is a lack of publicly available datasets for intrusion detection in IoT containing and comprehensively representing real-world traces from a diversity of devices, operating systems, underlying protocols, and diverse attack types [217]. A recent work towards filling this gap is by [175, 180] who collected the NBaIoT datasets out of real IoT devices infected with popular botnets such as Mirai and BASHLITE. Hence, in this chapter, we use the NBaIoT datasets to evaluate the proposed method. In addition, for further evaluation of our method on a diversity of

intrusion attacks, and to compare against state-of-the-art IDS, we have also used the recently published CICIDS2017 realistic intrusion detection datasets [229].

In what follows, we provide descriptions of the datasets to elaborate on the reasons of choosing these (Section 3.3.1). Then, we describe the optimal parameters selection for the codebook learning and TempoCode classification (Section 3.3.2). The performance of the proposed method is evaluated based on the metrics described in Section 3.3.3. The computing platform utilized to evaluate our proposed method is equipped with an Intel i7 CPU, 4 cores, 16GB RAM, CPU clock rate 1.8GHz.

3.3.1 Datasets Description

CICIDS2017 Datasets

The CICIDS2017 is a recently published collection of realistic intrusion detection datasets that has been collected to overcome the challenges and limitations of other popularly used IDS datasets that have been proposed prior to it. Since our objective is to detect various kinds of intrusion attacks, we were interested in this dataset for having realistic traces representing a variety of attacks. The CICIDS2017 dataset built by researchers from the Canadian Institute of Cybersecurity (University of New Brunswick), has several advantages (over other datasets used in the literature) besides the attack diversity, operating system variety, local and internet communications, etc., as discussed in detail in [229].

The dataset comprises of malicious traffic arising from six diverse attacks types: (i) Brute-force, (ii) Heartbleed, (iii) Botnet, (iv) DoS and DDoS, (vi) Web Attack, and (vii) Infiltration attacks. In Table 3.2, we provide a brief description of these attacks. As for the benign traffic recorded in the dataset, it is based on a realistic benign profiling system, covering a mix of protocols such as email, SSH, FTP, HTTP, HTTPS over TCP/UDP.

The Table 3.3 shows the composition of the dataset which was collected over a period of five days (Monday to Friday) during (nine) working hours each day. Different attacks were run on each day, leaving Monday with benign traffic only. As we can see, there is huge imbalance in the dataset due to the low ratio of malicious samples for many attack types. For example, the Friday morning dataset for Bot attacks has only about 0.01% of malicious

Table 3.2: Attack Types in CICIDS2017 Datasets

Attack Type	**Description**
Bruteforce	Based on the FTP- and SSH-Patator tools. The attacker tries to gain access to content or documents via a hit and try method
Heartbleed	Targeted against OpenSSL-based Transport Layer Security (TLS) protocol
Botnet	A number of devices are compromised and exploited to carry out different attacks/operations. Ares-based Botnet
DoS/DDoS	Targeted against a network resource or service to make it unavailable for benign users. When many different devices are exploited (e.g. by a botnet), it is called DDoS. Tools used: GoldenEye, Slowloris, Hulk, Slowhttptest, Heartleech, LOIC
Web Attack	Attacks like SQL Injection or Cross-Site Scripting (XSS), over the web, exploiting vulnerabilities in code
Infiltration Attack	Internally originated attacks. Attacker exploits software vulnerabilities to setup a backdoor on victim devices to carry out various attacks such as portscan or IP sweep, etc. Tool: Metasploit, Nmap, portscan

Table 3.3: Composition of CICIDS2017 Datasets, and our training, validation and testing splits

Dataset	**Class**	**Samples**	**Ratio**	**#Tr**	**#Val**	**#Te**
Monday	Benign	529919	1.0	529919	-	-
Tuesday	Benign	431813	0.969	259088	86362	86363
	FTP-Patator	7935	0.018	4761	1587	1587
	SSH-Patator	5897	0.013	3539	1179	1179
Wednesday	Benign	439683	0.636	263810	87937	87936
	DoS GoldenEye	10293	0.015	6176	2059	2058
	DoS Hulk	230124	0.333	138074	46025	46025
	DoS Slowhtptest	5499	0.008	3299	1100	1100
	DoS Slowloris	5796	0.008	3478	1159	1159
	Heartbleed	11	0.000016	7	2	2
Thursday Morning	Benign	168051	0.988	100831	33610	33610
	Brute Force	1507	0.009	905	301	301
	SQL Injection	21	0.0001	13	4	4
	XSS	652	0.004	391	131	130
Thursday Afternoon	Benign	288359	0.9999	173015	57672	57672
	Infiltration	36	0.0001	22	7	7
Friday Morning	Benign	188955	0.99	113372	37791	37792
	Bot	1956	0.01	1174	391	391
Friday Afternoon1	Benign	127292	0.445	7676	2558	2558
	PortScan	158804	0.555	95283	31761	31760
Friday Afternoon2	Benign	97686	0.433	58612	19537	19537
	DDoS	128025	0.567	76815	25605	25605

samples. Hence, to avoid overfitting in training our classifiers, we randomly downsample the benign data so as to retain 50% benign flows and 50% malicious flows in each dataset. Despite downsampling the benign data for training, the results of the proposed methods are encouragingly good in detecting the benign flows (see Section 3.4). In our experiments, we partition the respective datasets into 60% training, 20% validation, and 20% testing sets (represented by #Tr, #Val, and #Te in Table 3.3, respectively), following a popular approach in classification works [7]. We use the benign flows from Monday's traces to learn the codebook $\mathbf{C_T}$.

NBaIoT Datasets

The NBaIoT datasets of [175, 180] are the best IoT-related intrusion detection datasets publicly available, to best of our knowledge. The datasets were collected from a real testbed of nine wireless-ly connected IoT devices, setup in a way to resemble an enterprise setting. These devices were infected with two families of real-world IoT-based botnets (Mirai and BASHLITE/Gafgyt). The dataset contains the packet- and flow-based features representing the benign and malicious traffic.

Table 3.4 provides a summary of the different attacks covered in the NBaIoT dataset under the Mirai and BASHLITE botnet families. The composition of the dataset is given in Table 3.5. Following the approach of [175], we split the datasets (benign and malicious samples) into equal-sized partitions for training, validation and testing. The benign traffic data for each IoT device consists of frequent and infrequent actions as well.

3.3.2 TempoCode Configurations

The main parameters to configure TempoCode -based intrusion detection are the t_{dur} (length of time windows) and N_{ct} (number of codewords to learn from each time window). In this work, we utilise the CICIDS2017 datasets (due to the wide diversity of attack types it contains) in a binary classification setting (i.e., binary vs malicious) to study the effect of TempoCode parameters on the intrusion detection performance.

Table 3.4: Attack Types in NBaIoT Dataset

Attack Type	**Botnet Family**	**Description**
Scan	Mirai and BASHLITE (Gafgyt)	Looks for vulnerable devices in the network
Junk	BASHLITE (Gafgyt)	Sends junk/spam data
COMBO	BASHLITE (Gafgyt)	Sends spam data; Opens connection to a given IP address and port
Flooding	Mirai	ACK, SYN, UDP, UDPplain (higher PPS, enabled by fewer options)
	BASHLITE (Gafgyt)	UDP, TCP

Table 3.5: Composition of NBaIoT Datasets: IoT devices, and number of benign and malicious samples

ID	**Device**	**#Benign**	**#Mirai**	**#Gafgyt**
1	Danmini (Doorbell)	40395	652100	316650
2	Ecobee (Thermostat)	13111	512133	310630
3	Ennio (Doorbell)	34692	N/A	316400
4	Philips B120N10 (Baby Monitor)	160137	610714	312723
5	Provision PT737E (Security Camera)	55169	436010	330096
6	Provision PT838 (Security Camera)	91555	429337	309040
7	Samsung SNH1011N (Webcam)	46817	N/A	323072
8	SimpleHome XC57-1002-WHT (Security Camera)	42784	513248	303223
9	SimpleHome XC57-1003-WHT (Security Camera)	17936	514860	316438

Effect of t_{dur}

We evaluated TempoCode-based intrusion detection under varying t_{dur} from 0.25 to 1.0, in steps of 0.25. This respectively corresponds to taking time windows of 15 mins., 30 mins., 45 mins., and 1 hour, in the codebook learning phase. Figure 3.3(a) shows how correct classification rate (or ACC_i) varies with t_{dur}, while Figure 3.3(b) shows the average processing time (ms) per TempoCode vector during testing phase.

We note that the best ACC_i is obtained when t_{dur} is 0.25. However, this comes at a cost of higher processing time per TempoCode vector (around 15.57ms) in comparison to the lower processing times at higher t_{dur}. This is expected because with lower t_{dur}, the codebook size increases (given a fixed N_{ct}), and hence the dimensionality of TempoCode vectors increases.

An interesting observation is when $t_{dur} = 0.75$, the ACC_i is lower than at $t_{dur} = 1.0$, although we expected it to be higher. This indicates that with a time window of 45 mins., for the CICIDS2017 dataset, the learnt temporal codebook doesn't capture the benign patterns as effectively as with a time window of 1 hour.

The specific choice of t_{dur} would depend on an application's or system's requirements and constraints, taking into consideration a trade-off between accuracy and processing time. In this chapter, we choose $t_{dur} = 1$ which yielded the lowest processing time (2.94ms) and a reasonably good ACC_i of around 98.79%.

Effect of N_{ct} and $CSize$

The number of patterns N_{ct} learnt per time window in the codebook C_T plays a role in determining the performance of TempoCode in terms of accuracy as well as processing time. It is expected that at a higher N_{ct}, since the codebook is more comprehensive than at a lower N_{ct}, a higher accuracy could be achieved.

In Table 3.6, we present the classification scores of TempoCode with varying N_{ct} values (i.e., 5, 10, 15, 20, 25). As it can be observed, the precision, recall and F1 scores for both benign and malicious classes increased with increase in N_{ct}. The $CSize$ column shows the overall codebook size for each N_{ct}

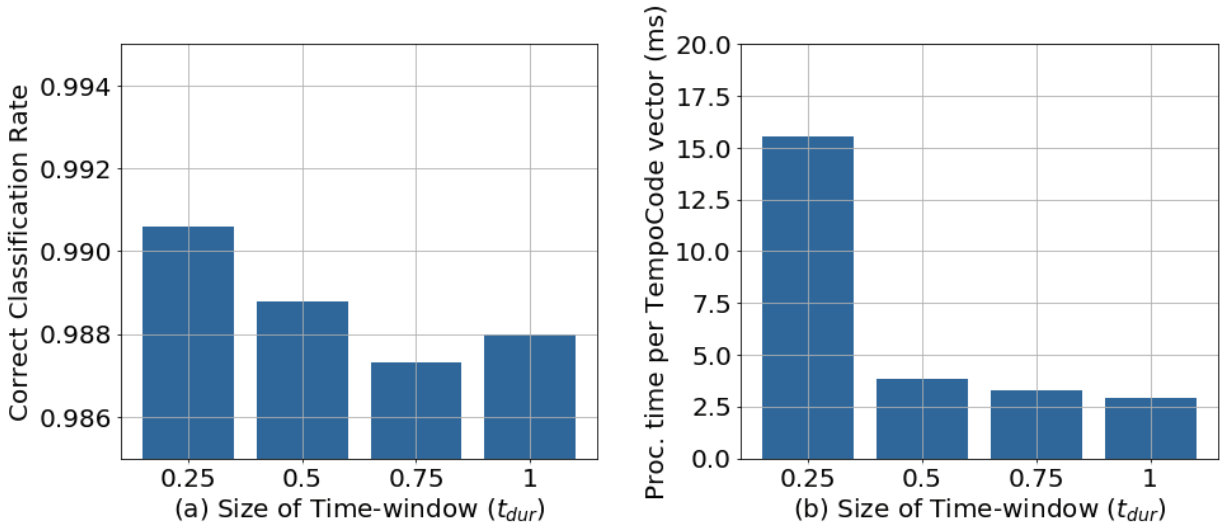


Figure 3.3: (a) Effect of t_{dur} on accuracy (correct classification rate of benign and malicious samples) and (b) Processing time (per TempoCode vector)

The best F1 scores for the benign and malicious classes were achieved at $N_{ct} = 15$. Looking at the precision of benign class (Prec_Ben), F1 scores of benign and malicious classes, as well as the recall of malicious class, the scores were better at $N_{ct} = 15$ than at $N_{ct} = 20$.

Table 3.6: Effect of N_{ct} and $CSize$ (Codebook Size) on TempoCode Classification Scores

$\mathbf{N_{ct}}$	**CSize**	**Ben-Ben**	**Ben-Mal**	**Mal-Ben**	**Mal-Mal**	$\mathbf{P_{ben}}$	$\mathbf{R_{ben}}$	$\mathbf{F1_{ben}}$	$\mathbf{P_{mal}}$	$\mathbf{R_{mal}}$	$\mathbf{F1_{mal}}$
5	45	345555	2813	2917	108394	0.9916	0.9920	0.9918	0.9747	0.9738	0.9743
10	90	346495	1873	2577	108734	0.9926	0.9946	0.9936	0.9831	0.9769	0.9800
15	135	346618	1750	**2534**	**108777**	**0.9927**	0.9950	**0.9939**	0.9842	**0.9772**	**0.9807**
20	180	**346762**	**1606**	2794	108517	0.9920	**0.9954**	0.9937	**0.9854**	0.9749	0.9801

From Table 3.6, we also note that a larger codebook (higher $CSize$, as a consequence of a higher N_{ct}) results in higher classification scores. These results are obtained by setting $t_{dur} = 1.0$, which makes $N_T = 9$ non-overlapping time windows (on Monday's benign flows dataset of CICIDS2017), where the $CSize = N_T \cdot N_{ct}$. However, at $N_{ct} = 20$, the F1

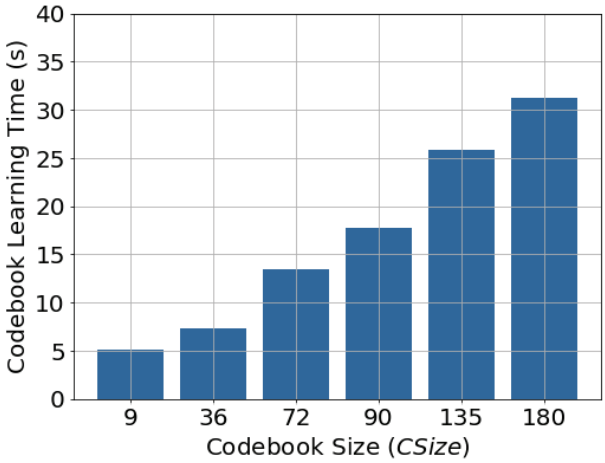


Figure 3.4: Codebook learning time for TempoCode, with varying $CSize$

scores are lower than those at $N_{ct} = 15$. A reason could be because the codebook becomes redundantly large at $N_{ct} = 20$ for the used dataset, leading to TempoCode representations which cause classifier confusion or to classifier overfitting, deduced from the observation that at $N_{ct} = 20$, Ben-Ben (TP) and Mal-Ben (FP) counts are higher than at $N_{ct} = 15$.

Codebook Learning Time

The time consumed in learning the codebook is an important factor in assessing the adaptability of the proposed method. Application scenarios in which the benign network traffic behavior may evolve or change could require re-training the codebook. Hence, having lower codebook learning times are beneficial for such cases. However, as we have shown above, smaller codebooks could fall short of capturing the benign flow patterns thereby reducing accuracies. In Figure 3.4, we show the codebook learning times of TempoCode for increasing $CSize$. The TempoCode's codebook size depends on N_{ct}. In this figure, we observed the codebook learning times for $N_{ct} = 1, 4, 8, 10, 15, 20$ corresponding to $CSize = 9, 36, 72, 90, 135, 180$, respectively (Note that the time-window $t_{dur} = 1$ hour and the CICIDS2017 dataset has $N_T = 9$ hours of data).

In learning the temporal codebook, for each time-window, the clustering process is applied only on the benign samples from that time-window, and not on the entire set of

samples over the whole duration of the dataset. This allows for parallel codebook learning, where each time-window's codebook could be learnt in parallel.

Classifier Configurations

In this work, for TempoCode-based intrusion detection, we used the following parameters for the ensemble of SVM classifiers. The ensemble size was 10, using a majority voting-based classification output. Based on empirical evaluations, we found that the constituent SVM classifiers performed best with $C = 2000$, $\gamma = 500$, for the CICIDS2017 datasets.

3.3.3 Performance Metrics

We assess the performance of the proposed TempoCode-based intrusion detection based on the following metrics: Precision, Recall, F1-score, and Class-wise accuracy. In addition, we utilise the confusion matrices as a tool to interpret and understand the performance in terms of these scores.

- Precision: For a class i, its Precision score P_i is:

$$P_i = \frac{TP_i}{TP_i + FP_i} \tag{3.5}$$

- Recall: For a class i, its Recall score R_i is:

$$R_i = \frac{TP_i}{TP_i + FN_i} \tag{3.6}$$

- F1-Score: a harmonic average of Precision and Recall scores

$$F1_i = \frac{2 \cdot P_i \cdot R_i}{P_i + R_i} \tag{3.7}$$

- Class-wise Accuracy (or Correct Classification Rate): A measure of how many samples were correctly classified (TP_i) as benign or corresponding attack class respectively, with respect to the total number of test samples of the class.

$$ACC_i = \frac{TP_i}{\#TestSamples_i} \tag{3.8}$$

- Confusion Matrix: Depicts the percentage of test samples of each ground-truth class (represented by rows) classified to each class (represented by columns). So, the main diagonal values are the ACC_i and R_i, while the other values indicate the False Positives or False Negatives. In a row i, the value at (i, i) is the ACC_i and R_i; the values in (i, j), where $j \neq i$, show the False Negatives (FN_i), i.e., the percentage of samples of class-i that were mis-classified to class-j.

The definitions of TP, FP, FN are based on the class being considered and the granularity of classification (binary or multi-class). While TP_i refers to the True Positives, FP_i refers to the False Positives, and FN_i refers to the False Negatives with respect to class-i. In a binary classification setting (Section 3.4.2), the two classes are 'Benign' ($i = 0$) and 'Malicious' ($i = 1$). The TPs with respect to benign class (TP_0) are the benign samples which were classified as benign, while the FPs with respect to benign class (FP_0) are the malicious samples which were classified as benign. Similarly, TP_1 and FP_1 refer to the TPs and FPs with respect to the malicious class. In multi-class classification (Section 3.4.1), we consider TP, FP, FN with respect to each class. So, TP_i would be the samples of class-i classified as class-i while TP_j would be samples of class-j classified as class-j. In calculating the ACC_i, P_i, R_i, $F1_i$ for a class-i, we consider the TP, FP, FN defined with respect to class-i.

3.4 Results & Discussions

To evaluate the performance of the proposed TempoCode representations for intrusion detection, we use the CICIDS2017 and NBaIoT datasets (described in Section 3.3.1). With the CICIDS2017 datasets, we conduct two sets of experiments. First, in Section 3.4.1, we evaluate TempoCode representations for detection of individual attack types in the different datasets of CICIDS2017 (as listed in Table 3.3). Second, in Section 3.4.2, we investigate the effectiveness of TempoCode representations in a binary classification setting (to differentiate between benign and malicious flows). On the NBaIoT datasets, we evaluate the performance of Tempo-Coderepresentations in differentiating the benign flows from the malicious flows arising from the compromised IoT devices. Finally, in Section 3.4.4,

we compare the performance of our proposed method with results of prior works on the CICIDS2017 and NBaIoT datasets, demonstrating the superiority of our method over state-of-the-art.

3.4.1 On the CICIDS2017 Datasets: Attack Type Classification

In these set of experiments, we evaluate the performance of TempoCode-based IDS on each dataset of CICIDS2017 [229], i.e., each attack type's dataset. Figures 3.5,3.6, 3.7,3.8 show the confusion matrices for each attack type. The results are summarised in Table 3.7 which presents the average scores along with the respective 95% confidence intervals from 10 test runs (i.e., 10 randomly split test subsets). The temporal codebook $\mathbf{C_T}$ is learnt using Monday's benign training data samples, with size $N_{ct} = 15$, $T_{dur} = 1$, $N_T = 9$, i.e., $CSize = 135$ used in these experiments, based on the findings in Section 3.3.2.

In the case of DoS attacks dataset, looking at Figure 3.5 and Table 3.7, we observe the following. On average, 99.34% of benign samples were correctly classified as benign, with the benign class achieving an average f1-score of 0.9989 ± 0.0005. The four variants of DoS attacks, namely the GoldenEye, Hulk, Slowhttptest, and Slowloris also achieved high ACC_i and recall scores. However, the Heartbleed samples were mis-classified as benign, and hence the poor performance scores. The total number of samples in the dataset for the Heartbleed class were only 11, out of which 60% (i.e., 7) were taken for training, 20% (i.e., 2) for validation, and 20% (i.e., 2) for testing. Due to the large imbalance between the number of samples of benign, other DoS variants, and Heartbleed classes, the classifier wasn't able to perform well on the Heartbleed samples. Another observation to make here are the confusions between the DoS variants. For example, 0.91% of Slowhttptest samples were confused to be of Slowloris. This indicates some degree of similarity in the feature space, between the samples from DoS attack variants.

On the web attacks dataset, we note that the benign and bruteforce web attack samples were very accurately classified, at average recall scores of 0.9911 ± 0.0010 and 0.9928 ± 0.0109, respectively (see Table 3.7). However, the SQL injection and XSS variants of web attacks suffered from very low classification scores. Observing the confusion matrix (Figure 3.6(Left)) gives an insight to the cause. As one can see, 75% of the SQL injection

attack samples were mis-classified as bruteforce web attack, and 57.14% of XSS attack samples were also mis-classified as bruteforce web attacks. Although these samples were not classified to the correct attack variant, they were correctly identified as non-benign, thereby maintaining a high intrusion detection performance.

Looking at the results on Bruteforce attacks dataset based on the FTP- and SSH-Patator tools, the average f1-scores achieved for the benign, FTP-Patator, and SSH-Patator classes were 0.9989 ± 0.0005, 0.9127 ± 0.0134, and 0.7574 ± 0.0184, respectively (see Table 3.7). The confusions between these classes were also very low (see Figure 3.6 [Right])

In the case of Infiltration attacks, TempoCode's performance was not as high as expected (see Figure 3.7 [Left] and Table 3.7). Although the benign samples were correctly classified at an average recall of 0.9994 ± 0.0002, a high percentage of infiltration attack samples were mis-classified as benign. This could be attributed to the insufficient number of training samples for the Infiltration classes, as mentioned in Table 3.3.

On the Portscan attacks dataset, the achieved average f1-scores were 0.9992 ± 0.0003 and 0.9993 ± 0.0002 for the Benign and Port-scan classes, respectively. Moreover, the false positives count (i.e., malicious samples mis-classified as benign samples) were very low, around 0.08% (see Figure 3.7 [Right]) Similarly encouraging results were obtained on the Botnet and DDoS attacks dataset (see Figure 3.8), where the Botnet attacks were detected at an average recall of 0.9600 ± 0.0192, and the DDoS attacks at an average recall of 0.9928 ± 0.0013.

3.4.2 On the CICIDS2017 Datasets: Binary Classification (Benign vs Malicious)

In these set of experiments, we investigate how well can TempoCode representations discriminate between benign samples and the diversity of malicious samples wherein the samples from various attack types from the individual datasets were combined together and labelled as "malicious".

We utilise the codebook $\mathbf{C_T}$ (which was learnt from Monday's benign training data, with parameters described in Section 3.3.2) to generate TempoCode representations respectively. The samples from rest of the days were combined together in one dataset and

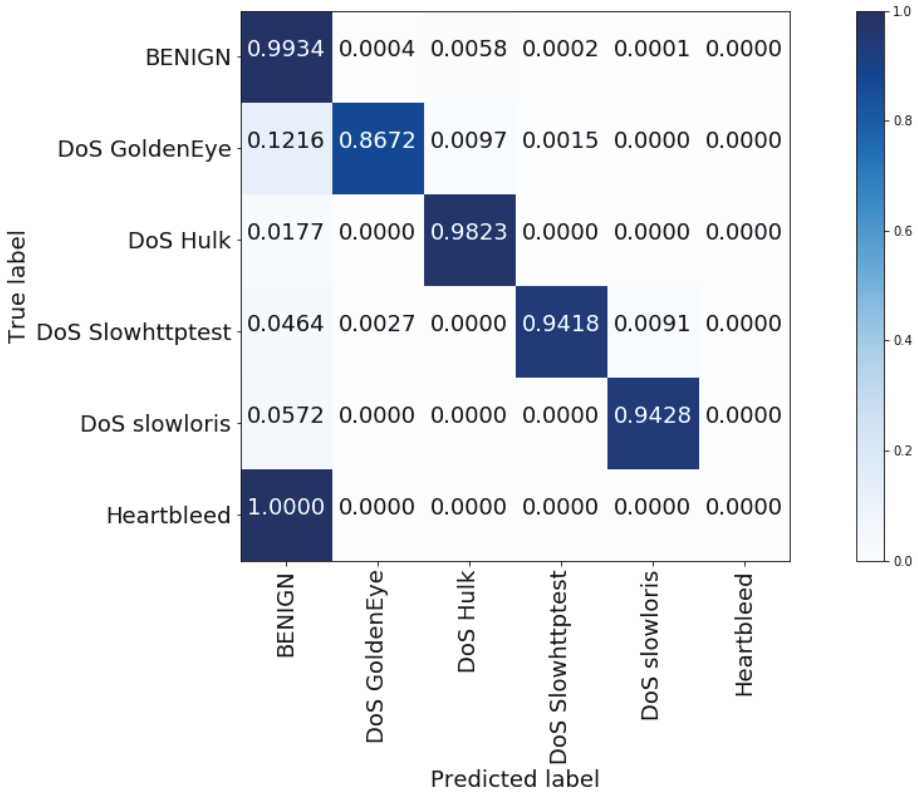


Figure 3.5: TempoCode: Benign vs DoS (GoldenEye, Hulk, Slowhttptest, Slowloris, Heartbleed)

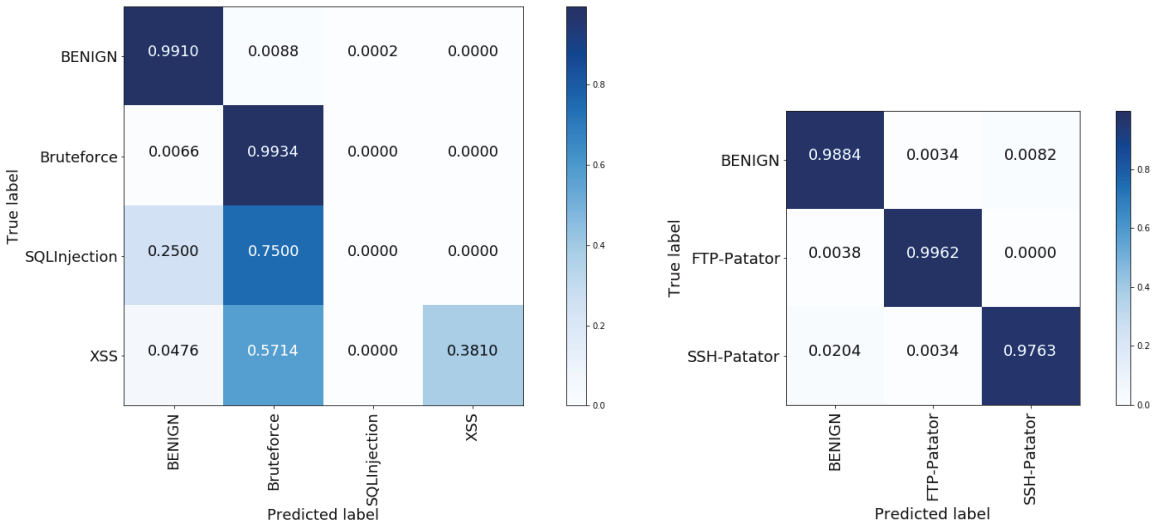


Figure 3.6: TempoCode: [Left] Benign vs Web Attacks (BruteForce, SQL Injection, XSS); [Right] Benign vs Patator-based Bruteforce attacks over FTP/SSH

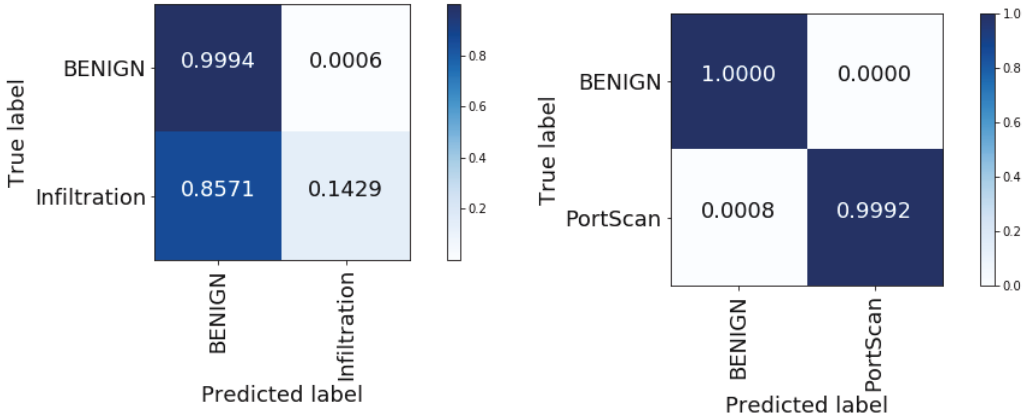


Figure 3.7: TempoCode: Benign vs Infiltration, and Benign vs PortScan

Table 3.7: Performance Evaluation of TempoCode-based Intrusion Detection on CICIDS2017 Datasets

Dataset	**Class**	**Precision**	**Recall**	**F1-Score**
Tuesday	Benign	0.9996±0.0001	0.9989±0.001	0.9989±0.0005
	FTP-Patator	0.8427±0.0220	0.9961±0.0034	0.9127±0.0134
	SSH-Patator	0.6194±0.0246	0.9763±0.0094	0.7574±0.0184
Wednesday	Benign	0.9866±0.0007	0.9934±0.0007	0.9900±0.0006
	DoS GoldenEye	0.978±0.0064	0.866±0.0104	0.9184±0.0060
	DoS Hulk	0.9884±0.001	0.9823±0.001	0.9853±0.001
	DoS Slowhtptest	0.9717±0.012	0.9429±0.018	0.9567±0.008
	DoS Slowloris	0.9832±0.012	0.9374±0.009	0.9596±0.005
	Heartbleed	0.0	0.0	0.0
Thursday Morning	Benign	0.9996±0.0002	0.9911±0.0010	0.9953±0.0005
	Brute Force	0.4174±0.0184	0.9928±0.0109	0.5873±0.0109
	SQL Injection	0.0	0.0	0.0
	XSS	0.0	0.0	0.0
Thursday Afternoon	Benign	0.9999±0.0001	0.9994±0.0002	0.9997±0.0001
	Infiltration	0.0333±0.0533	0.0667±0.0107	0.0444±0.0711
Friday Morning	Benign	0.9585±0.0197	0.9742±0.0191	0.9659±0.0129
	Bot	0.9736±0.0205	0.9600±0.0192	0.9663±0.0124
Friday Afternoon1	Benign	0.9985±0.0005	0.9998±0.0002	0.9992±0.0003
	PortScan	0.9999±0.0001	0.9988±0.0005	0.9993±0.0002
Friday Afternoon2	Benign	0.9907±0.0017	0.9983±0.0005	0.9945±0.0008
	DDoS	0.9987±0.0004	0.9928±0.0013	0.9958±0.0007

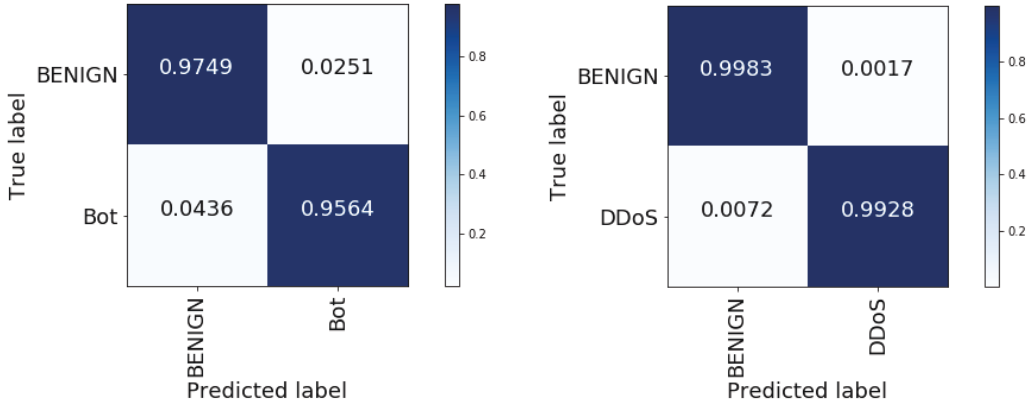


Figure 3.8: TempoCode: Benign vs Bot, and Benign vs DDoS

subsequently partitioned into training, validation and testing splits following the 60%-20%-20% ratio. The best performing $CSize = 135$ is taken for the codebook (see Section 3.3.2). Consequently, the proposed scheme is referred to as TempoCode-135. In binary classification, the objective is to detect anomalous (malicious) samples which may belong to different attack types and hence different characteristics.

Figure 3.9 shows the confusion matrix between benign and malicious classes for the Tempo-Code-based scheme. On average, around 99.50% of benign samples and 97.72% of malicious samples were correctly classified. The mean False Positive Rate was 2.28% while the mean False Negative Rate was 0.5%. The precision, recall and f1 scores are provided in Table 3.8, along with the 95% confidence intervals obtained from tests on 10 randomly partitioned subsets of the test data.

Table 3.8: Binary Classification Scores with TempoCode on CICIDS2017 Datasets

Class	**Precision**	**Recall**	**F1-Score**
Benign	0.9927±0.0004	0.9950±0.0003	0.9939±0.0003
Malicious	0.9842±0.0010	0.9772±0.0014	0.9807±0.0011

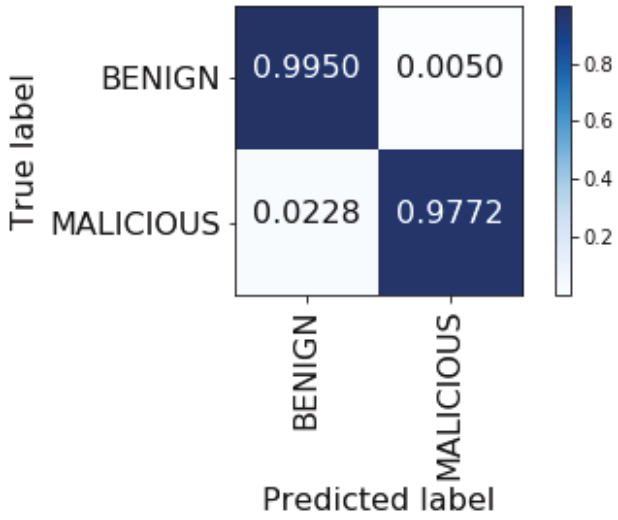


Figure 3.9: Confusion Matrix for TempoCode-based Binary Classification

3.4.3 On the NBaIoT Datasets: Botnet Attacks Detection

In our empirical evaluations of TempoCode on the NBaIoT datasets, we adopt the following approach. Considering the diversity of device types, we propose to learn a temporal codebook for each device, capturing its key benign flow patterns. Each device's dataset is split into three equal-sized partitions for training, validation and testing (similar to the approach of [175]). The features (as described in Section 3.2.1) in each device's dataset were normalized to be in the range $[0, 1]$, and only the benign traffic's features from the training dataset were used in codebook learning for the device.

Due to the lack of timestamp data in the datasets, we adaptively set the size of time-window (t_{dur}) depending upon the number of benign training samples available for a device. For each device, t_{dur} is taken as a number of samples given by $\frac{N_{tr-ben}}{(CSize/N_{ct})}$, where N_{tr-ben} is the number of benign samples in training dataset of this device, $CSize = 140$ (codebook size), and $N_{ct} = 10$ (number of codewords to be learnt from each time-window). For example, the first t_{dur} number of samples could have arrived in say 1 hour while the next t_{dur} number of samples could have arrived in 0.5 hour. Our empirical evaluations have proven the strength of TempoCode representations even with such an adaptive t_{dur}. The values of $CSize$ and N_{ct} have been chosen such that the learnt codebook is of $CSize = 140$

for all devices, to be close to the best codebook size found in TempoCode evaluations on CICIDS2017 datasets (as described in Section 3.3.2).

Consequently, the TempoCode representations of the benign and malicious flows from each device are generated using the respective device's temporal codebook. Based on these TempoCode representations, the ensemble of SVM classifiers is trained using the training dataset and optimized using the validation dataset, in a supervised manner. Because each device has different behaviors and actions resulting in different traffic behaviors, a separate ensemble of SVM classifiers is learnt for each device (see Table 3.9). The testing datasets which contain a mix of benign and malicious samples that have not been seen in the training phase are used to evaluate the performance of TempoCode representations through the ensemble of SVMs. Each test set is further split into 10 random partitions to run 10 tests for each device and obtain average scores with 95% confidence intervals.

The experimental evaluations of the proposed Tempo-Code-IoT-based intrusion detection yielded very encouraging results for all the devices (see Figure 3.10). The proposed method was able to differentiate between benign and malicious traffic from each compromised IoT device with very high recall scores (0.9940-0.9991), and very low false positive rates ($0.02 - 0.71\%$), as observed from Table 3.10 which presents the scores averaged over 10 test runs along with the respective 95% confidence intervals.

These results indicate the effectiveness of the proposed temporal codebook in learning key benign traffic patterns of each device and the high discriminative capability exhibited by the TempoCode representations through the ensemble of SVM classifiers. Moreover, since our approach builds a temporal codebook and ensemble of classifiers for each device (using the flows from the respective device only), the addition of new IoT devices would only require learning of a codebook and classifier ensembles for those specific devices alone. In this way, re-training costs are avoided as the system doesn't require re-training the codebooks and classifiers previously learnt for other devices. Furthermore, such an approach yields tolerance to a growing heterogeneity of IoT devices.

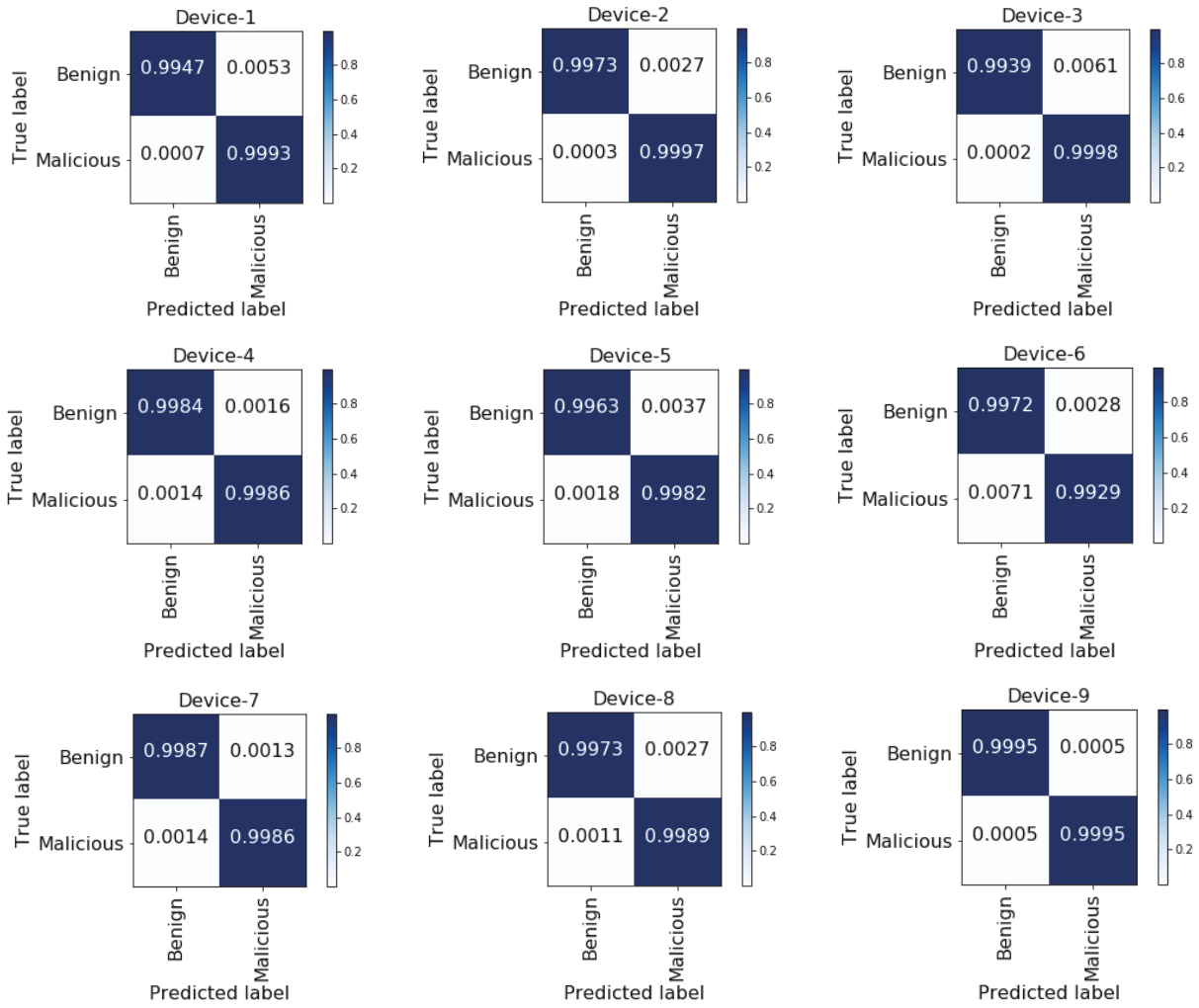


Figure 3.10: Evaluating TempoCode on NBaIoT Datasets: Confusion matrices for each of the nine devices (the values are averaged across 10 test runs)

Table 3.9: TempoCode Evaluations on the NBaIoT Datasets: SVM Parameters, Number of training/testing samples ($N_{tr}, N_{tr-ben}, N_{te}$), Codebook Learning Time (CLT), Time consumed in training/testing (T_{tr}, T_{te})

Dev	**SVM Params**	$\mathbf{N_{tr-ben}}$	**CLT (s)**	$\mathbf{N_{tr}}$	$\mathbf{T_{tr}}$ (s)	$\mathbf{N_{te}}$	$\mathbf{T_{te}}$ (s)
1	C1000, G0.001	13465	4.24	26930	2.31	336389	14.1
2	C1000, G0.001	4370	2.15	8740	5.29	278619	10.5s
3	C1000, G0.001	11564	2.21	23128	1.5	117034	4.4
4	C0.1, G0.001	53379	9.03	106758	10.5	361198	114
5	C1, G0.001	18389	2.34	36778	2.67	273769	35.8
6	C10, G0.001	30518	3.38	61036	3.1	276652	19.5
7	C10, G0.001	15605	3.21	31210	2.08	123301	5.7
8	C10, G0.001	14261	4.81	28522	2.34	286423	27.6
9	C100, G0.01	5978	1.32	11956	1.4	283086	10.7

Table 3.10: TempoCode Results on the NBaIoT Datasets: Precision, Recall, F1 scores for benign and malicious classes, and False Positive Rate (FPR)

Device	$\mathbf{P_{ben}}$	$\mathbf{R_{ben}}$	$\mathbf{F1_{ben}}$	$\mathbf{P_{mal}}$	$\mathbf{R_{mal}}$	$\mathbf{F1_{mal}}$	**FPR (%)**
1	0.9845±0.0016	0.9948±0.0011	0.9896±0.0010	0.9998±0.0	0.9993±0.0001	.9996±0.0001	0.07±0.01
2	0.9829±0.0027	0.9973±0.0010	0.9900±0.0013	1.0±0.0	0.9997±0.0001	0.9998±0.0	0.03±0.0
3	0.9983±0.0009	0.9940±0.0019	0.9961±0.0011	0.9993±0.0002	0.9998±0.0001	0.9996±0.0001	0.02±0.01
4	0.9919±0.0007	0.9984±0.0004	0.9951±0.0004	0.9997±0.0001	0.9986±0.0001	0.9992±0.0001	0.14±0.01
5	0.9754±0.0025	0.9963±0.0015	0.9858±0.0015	0.9997±0.0001	0.9982±0.0002	0.9990±0.0001	0.18±0.02
6	0.9455±0.0027	0.9972±0.0006	0.9706±0.0013	0.9997±0.0001	0.9929±0.0003	0.9963±0.0001	0.71±0.03
7	0.9903±0.0021	0.9987±0.0006	0.9945±0.0010	0.9998±0.0001	0.9986±0.0003	0.9992±0.0002	0.14±0.03
8	0.9800±0.0026	0.9973±0.0010	0.9886±0.0013	0.9999±0.0001	0.9989±0.0001	0.9994±0.0001	0.11±0.01
9	0.9779±0.0036	0.9991±0.0009	0.9889±0.0019	1.0±0.0	0.9995±0.0001	0.9997±0.0001	0.05±0.01

3.4.4 Comparison with Related Works

In this section, we present a comparison of TempoCode's performance against other methods proposed recently on the CICIDS2017 and NBaIoT datasets. Table 3.11 provides the precision, recall and f1 scores (for detection of malicious flows) of our work and those reported by the respective related works on the CICIDS2017 datasets. Additionally, Table 3.12 compares our results with those reported in [175] on the NBaIoT datasets.

On the CICIDS2017 Dataset

Marir et al. [172] proposed a deep belief network (DBN) and a multi-layer ensemble of SVMs (MLE-SVM) for detection of malicious flows. They studied the performance of DBN and MLE-SVM individually, as well as in a coupled fashion. The proposed TempoCode method of this paper outperforms the three methods of [172].

To evaluate the performance of the proposed Tempo-Code method against the raw statistical flow features (such as those mentioned in Section 3.2.1), we compare with the top three scores reported in [229]'s study. In their work, they found the following three machine learning algorithms performed best in detecting the malicious flows: k-Nearest Neighbors (kNN), Random Forest, and ID3, based on the raw statistical flow features. From Table 3.11, we can see that TempoCode performed better than their kNN and RF methods and slightly better than the ID3-based method. The results of [229] were after a feature selection process by which only the most important features were retained. If a similar procedure is applied to TempoCode features, it could result in even better performance. However, we dedicate this to be investigated in a future work.

On the NBaIoT Datasets

We compare the performance of TempoCode against the results of two recently published works on NBaIoT data-sets: [187] and [175]. In comparison to the results of deep auto-encoders-based intrusion detection of [175] who achieved a mean FPR of 0.7%, the proposed Tempo-Code yielded a slightly higher mean FPR of 0.16% (still below 1%). In addition, their work tested other methods such as Isolation Forest and Local Outlier Factor (LOF)

Table 3.11: Performance Comparison of TempoCode with Related Works on CICIDS2017 Datasets

Method	**Precision**	**Recall**	**F1-Score**
DBN [172]	0.9006	0.9540	0.9265
MLE-SVM [172]	0.9056	0.9494	0.9269
DBN with LE-SVM [172]	0.9040	0.9565	0.9295
kNN [229]	0.9600	0.9600	0.9600
RF [229]	0.9800	0.9700	0.9700
ID3 [229]	0.9800	**0.9800**	0.9800
TempoCode (Our)	**0.9842**	0.9772	**0.9807**

Table 3.12: Performance Comparison of TempoCode with results reported by [187] on NBaIoT Datasets, in terms of Precision scores (malicious class)

Work	**Device**								
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
[187]	0.9952	0.9981	N/A	0.9474	0.9914	0.9860	N/A	0.9938	0.9976
Tempo-Code (Our)	0.9998	1.0	0.9993	0.9997	0.9997	0.9997	0.9998	0.9999	1.0

Table 3.13: Performance Comparison of TempoCode with results reported by [175] on NBaIoT Datasets, in terms of Mean False Positive Rates (FPR)

Method	**Mean FPR (%)**
Deep-Autoencoders	0.7
Isolation Forest	2.7
LOF	8.6
TempoCode (Our)	0.16

which also yielded higher mean FPRs of 2.7% and 8.6% respectively, as summarized in Table 3.13. Moreover, the mean detection time reported by [175] of 174ms is significantly slower than that of the proposed Tempo-Code-based method which required an average of 0.110ms per Tempo-Code vector.

Another work based on the NBaIoT dataset is of [187] who investigated SVMs and Isolation Forests trained over a subset of the statistical features provided by the datasets. The features were selected based on metrics such as entropy, variance and Hopkins statistic. As shown in Table 3.12, the proposed TempoCode-based method outperforms the best models of [187] for all devices. Their paper did not report results for devices 3 and 7.

The works of [175,187] are unsupervised learning-based approaches where the malicious traffic samples are detected as anomalies from the learnt benign traffic patterns. And although this paper proposed unsupervised learning of the temporal codebook to capture benign traffic patterns, the ensemble of classifiers is trained in a supervised fashion. The higher FPRs and lower precisions of [175,187] in comparison to our work indicate the benefits of supervised learning. However, considering the envisioned dynamic IoT ecosystem of tomorrow and evolving techniques of malicious attackers, we believe that unsupervised or possibly hybrid approaches may be a better choice. Hence, as part of our future work, we shall investigate TempoCode-based unsupervised anomaly detection approaches to identify intrusion attacks.

3.5 Concluding Remarks

In this chapter, we proposed *TempoCode*, a temporal codebook-based encoding of flow features, as a novel feature transformation of network flow features based on capturing the key patterns of benign traffic in a learnt temporal codebook. Using the codebook, distances of (benign and malicious) traffic flows from those key patterns are measured and recorded as the transformed features to train an ensemble of SVM classifiers. The experimental evaluations on recent realistic datasets (CICIDS2017 and NBaIoT) proved the effectiveness of TempoCode representations in detecting intrusions of various types and for different devices used in the datasets. On the NBaIoT datasets, TempoCode achieved high mean precision scores (0.9993 to 1.0), low mean False Positive Rates (0.02%

to 0.71%), and low average detection time of 0.110ms per TempoCode sample. Similarly, on the CICIDS2017 datasets, TempoCode achieved high precision and F1-scores of 0.9842 and 0.9807 respectively.

For future work, we plan to investigate the performance of TempoCode representations over larger and combined datasets of different attack types, in an unsupervised fashion, which effectively turns into a more challenging multi-class anomaly detection problem.

Chapter 4

Adaptive Ensembles of Autoencoders for Unsupervised IoT Network Intrusion Detection

Neural networks-based autoencoders have gained popularity, in the recent years, in problems of anomaly detection. Recent approaches have proposed ensembles of autoencoders to detect network intrusions. The computationally expensive ensembles of autoencoders make it challenging to be used for intrusion detection in networks of devices with lower resources (such as the Internet of Things) than in the cloud or data centers. To overcome this challenge, in this chapter, we propose, investigate and compare four methods to reduce the ensemble complexity through adaptive de-activations of autoencoders. These methods differ in their approach to select the autoencoders to de-activate (criteria-based or random) and differ when they conduct the de-activations (post-training or in-training). Extensive experiments on two recent, realistic IoT intrusion detection datasets validate the effectiveness of the proposed methods in achieving satisfactory detection performance at much lower training, re-training and inference time costs. The proposed methods shall enable scalable and efficient intrusion detection systems or services that could be deployed on-device or on-edge.

4.1 Introduction and Background

The networks are evolving with a growing diversity of inter-connected devices and things, advancing the landscape of an Internet of Things (IoT). Ranging from devices such as indoor or outdoor surveillance cameras, mobile user-worn devices such as smart watches or health-monitors, electrical and mechanical appliances, to connected vehicles and vehicular components, industrial systems, and connected smart cities, the IoT landscape is continuously evolving.

Along with the advancements in various connected things, services and applications, new vulnerabilities arise which continue to be exploited by malicious parties [98, 127, 130, 181, 191]. This poses danger not only to the devices but also to life and property. A serious example is of the Mirai botnet-based attack which exploited IoT devices to attack many popular web-based services and platforms that became inaccessible [147]. The targets of such DDoS attacks could include critical infrastructure, banks, healthcare institutions, smart cities and internet of connected vehicles and things [163, 176, 177, 179, 195, 217, 224, 228, 231, 259]. For example, connected vehicles have been shown to be vulnerable to being controlled by a remote malicious attacker who could shut down a moving car or lock/unlock doors [195]. In another worrisome example, smart (and connected) toys were found to have security and privacy flaws that could be exploited by an adversary to maliciously control the toys or cause serious privacy infringements [231].

In light of these incidents and the rise of novel threats, researchers in academia and industry are gearing efforts to develop innovative solutions for intrusion detection in IoT, to secure IoT from different types of intrusions [138, 180, 191, 208].

Recently, neural networks-based models such as Autoencoders (AEs) have received attention to address the problem of intrusion detection. AEs form a class of self-supervised learners that generate targets from the given inputs. Popularly implemented using neural networks, AEs learn to compress and decompress data through compression and decompression functions that are self-taught rather than hand-crafted by humans. Based on the above feature of AEs, they have found applications in various problems related to anomaly detection, for example, medical image anomaly detection of cancerous cells and network intrusion detection. In brief, AEs are used to capture the latent representations and be-

haviors of benign or normal data, with the assumption that malicious or anomalous data samples would lead to higher reconstruction errors.

The autoencoders considered in this work are based on artificial neural networks that are designed to enable learning latent representations of data. The distinguishing aspect in the design of autoencoders is that the intermediate layer(s) are of a lesser number of neurons, serving as an information bottleneck. The bottleneck layers ensure that the network does not simply memorize and pass the input values towards the output. The bottleneck layers effectively steer the network to learn compressed encodings and decodings of input data.

The autoencoders are tasked to reconstruct the input data. In order to train them to do so, a common approach is to minimize the reconstruction error (e.g., Root Mean Squared Error or RMSE) between the inputs $x_i \in X$ (where X represents some data) and the autoencoder outputs $\hat{x_i}$. The reconstruction error is backpropagated to update the weights of the autoencoder following Gradient Descent or Stochastic Gradient Descent methods (refer to [242] for more details on backpropagation). Given input data of n dimensions (i.e., features), the RMSE between input x_i and output $\hat{x_i}$ vectors is computed as follows:

$$RMSE(x_i, \hat{x_i}) = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \hat{x_i})^2}{n}} \tag{4.1}$$

Given training data, an autoencoder is trained as described above. Subsequently, in a basic autoencoder, using the reconstruction RMSEs from the trained autoencoder fed with benign input data samples, appropriate threshold(s) are determined such that a reconstruction error above this threshold would indicate a highly probable malicious or anomalous sample with respect to the training data's distribution.

Although recent works have proposed using AEs, ensembles of AEs [180], or stacked AEs [274], to address network intrusion detection problems, these methods remain computationally expensive for IoT devices. The major contribution of this chapter is introduction of four methods to reduce the complexity of AE-ensembles while yielding high detection performance at low training, re-training and inference time costs. The methods are based on two approaches to select AEs to de-activate in the ensemble: (i) *Criteria-based* and (ii) *Random*. In implementing these approaches, we propose two methods of choosing when to de-activate the selected AEs: *Post-Training* and *In-Training*. As the names suggest, the

former conducts de-activations of an ensemble's AEs after they have been trained, whereas the latter conducts de-activations during the training process itself, thereby making the training process adaptive and efficient as well.

The rest of the chapter is organised as follows. A brief review of relevant state-of-the-art works is presented in Section 2.2.2, followed by introducing and describing the proposed methods in Section 4.2. In Section 4.3, we describe the experimental setup, datasets used, and performance metrics used to evaluate the proposed methods. The results and discussions are presented in Section 4.4 and the chapter finally concludes in Section 4.5.

4.2 Proposed Methods: Adaptive Ensembles of Autoencoders

In this section, we start by providing an overview of the methods proposed in this chapter. This is followed by describing the network packets/flow-based features and feature subspaces over which the autoencoders are trained. A description of the ensemble of autoencoders is then presented, followed by describing the proposed methods of de-activations to achieve adaptive and efficient ensembles of AEs.

4.2.1 Overview

The proposed methods relate to an ensemble of autoencoders that learns the compressed latent representations of benign traffic data. In doing so, each autoencoder learns over a certain feature subspace instead of the entire feature space, to keep the complexity low.

We propose four methods to reduce ensemble complexity. These are based on two approaches to select which AEs to de-activate (*Criteria-based* and *Random*). To implement each of the two approaches, we propose two methods to decide when to conduct the de-activations (*Post-Training* and *In-training*). In the post-training de-activation method, autoencoders are only deactivated during the testing/inference phase. On the other hand, with the in-training de-activation method, autoencoders are deactivated during the training phase resulting in a final ensemble of a smaller size and complexity. Thus, the four

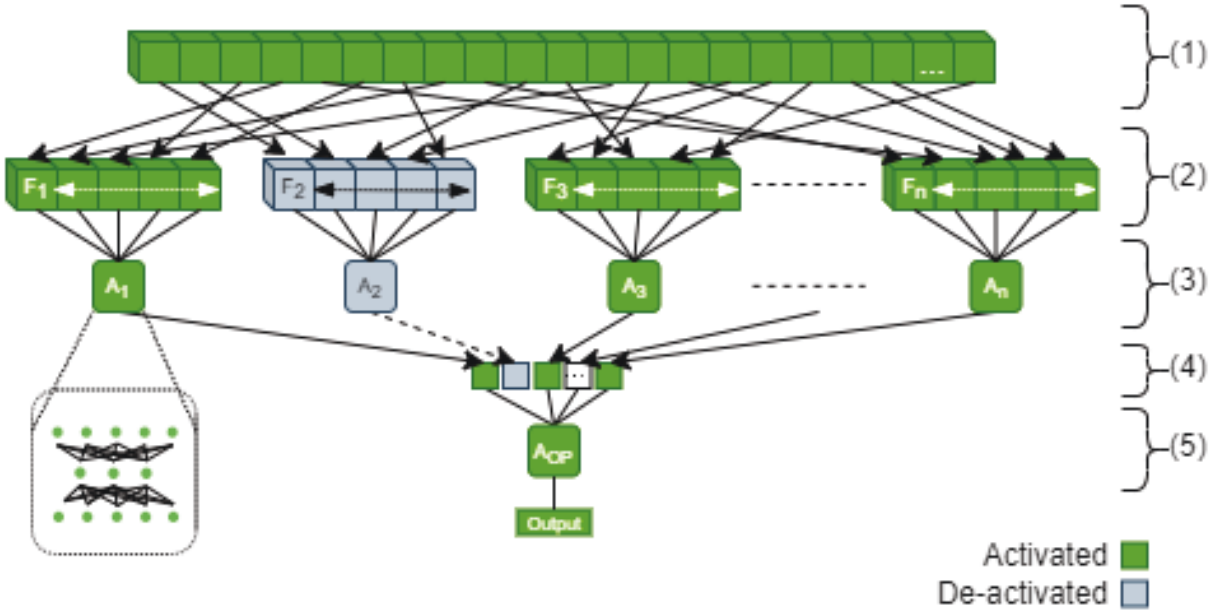


Figure 4.1: An illustrative representation of the proposed de-activations-based Adaptive Ensembles of Autoencoders. (1) indicates the input samples, (2) depicts splitting the input samples into the feature subspaces $\mathbf{F_i}$, (3) the ensemble of autoencoders, $E = \{A_i | i = 1, ..., n\}$ (where $n = N_{FS}$, the number of feature subspaces), (4) the collection of scores (e.g., RMSEs) from the A_i's, (5) the score aggregation module composed of the final output layer autoencoder, A_{OP}.

Table 4.1: Symbols and Notations (for Chapter 4)

Symbol	Description
CPTD	Criteria-based Post-Training De-Activation
CITD	Criteria-based In-Training De-Activation
RPTD	Random Post-Training De-activation
RITD	Random In-Training De-activation
AE	Auto-Encoder
N_F	Number of features
$\mathbf{F}$	The N_F-dimensional feature space
$dim_{\mathbf{F_i}}$	The dimensionality of feature subspace $\mathbf{F_i}$
F_j	A set of features corresponding to j-th sample
E	A set representing an ensemble of AEs (A_i's)
N_{FS}	Number of feature subspaces formed; the number of AEs in the original E
A_{OP}	The final output layer AE
C_d^r	A set of candidate AEs to de-activate, in round r
$\|C_d^r\|$	Size of the set C_d^r, in round r
E^r	The set representing the current ensemble in round r
$\|E^r\|$	Size of the set E^r

proposed methods in this work are: (i) Criteria-based Post-Training De-activations, (ii) Criteria-based In-Training De-activations, (iii) Random Post-Training De-activations, and (iv) Random In-Training De-activations, which shall be further described subsequently. Table 4.1 provides a description of the main symbols and notations used in this work.

4.2.2 Features and Subspaces

In the proposed methods, each autoencoder learns over a certain feature subspace instead of the entire feature space, to keep the complexity low. The set of features employed in this work, following the approach of Kitsune [180], include the following 23 statistics:

- *Packet Size (Outbound Traffic)*: Mean and variance of the outbound traffic band-

width aggregated by source MAC-IP, source IP, source-destination IP pair, source-destination TCP/UDP socket pair. (8 features)

- *Packet Count (Outbound Traffic)*: Aggregated by source MAC-IP, source IP, source-destination IP pair, source-destination TCP/UDP socket pair. (4 features)
- *Packet Jitter (Outbound Traffic)*: Mean, variance and count of inter-packet delays, aggregated by source-destination IP pair. (3 features)
- *Packet Size (Outbound and Inbound Traffic)*: Magnitude, radius, covariance, correlation coefficient of inbound and outbound traffic bandwidth together, aggregated by source-destination IP pair, source-destination TCP/UDP socket pair. (8 features)

These 23 statistics are extracted from the packets over five different time windows in the past (100ms, 500ms, 1.5s, 10s and 60s), yielding a set of $N_F = 115$ features in total. From each j-th sample in a dataset, the set of features F_j, is represented as $F_j = \{f_{1j}, f_{2j}..., f_{nj}\}$ (where $n = N_F$ represents number of features). For example, f_{1j} could represent the mean packet size from source-IP of j-th packet over a 100ms time window.

In this work, feature subspace selection is done deterministically as opposed to the feature-correlation based subspace selection used in other works such as Kitsune [180], thereby avoiding the time and computations required to build the feature subspaces. Figure 4.1 illustrates how the original feature space of the data is split into multiple subspaces. The feature space $\mathbf{F} \in R^N$ is split into N_{FS} number of subspaces $\mathbf{F_i}$ $(i = 1, ..., N_{FS})$, each of dimension say $dim_{\mathbf{F_i}}$.

4.2.3 Ensemble of Autoencoders

As described in Section 4.1, Autoencoders (AEs) are a class of neural networks that are capable of learning to reconstruct data. When trained in an unsupervised fashion, with benign data only, an AE is forced to learn the latent representations of benign traffic's features. The intuition behind using AEs is that, when they are trained on benign data only, any malicious sample fed into the AE should result in a badly reconstructed sample (i.e., with a high reconstruction error) when compared to that of a benign sample.

In the proposed methods, an ensemble of autoencoders represented by $E = \{A_i | i = 1, ..., N_{FS}\}$ is learnt such that an autoencoder A_i is trained over feature subspace $\mathbf{F_i}$ only (see Figure 4.1). Once trained, the ensemble outputs scores (one from each A_i) which represent the reconstruction errors, i.e., root mean square errors (RMSEs), of a given sample in the respective subspaces. The final AE in the output layer (represented by A_{OP}) is trained using the RMSEs from active ensemble layer AEs only. The final output layer A_{OP} learns the patterns in benign samples' RMSEs (and their relationships) produced by the ensemble layer AEs. At inference time, a sample is fed through the active AEs, obtaining a set of RMSEs from the active AEs which are then fed to the output layer. The output layer A_{OP} would then produce a reconstruction RMSE which would be used to decide if the sample is benign or malicious based on a certain threshold. In this way, A_{OP} produces a prediction of whether a given sample is benign or anomalous.

In brief, the A_{OP} learns over the reconstruction errors of benign samples such that in the testing phase, given a malicious test sample, A_{OP}'s reconstruction error is expected to be discriminable (in comparison to the reconstruction errors of benign samples) hence enabling identification of the sample as malicious. In other words, the RMSE from the output layer's AE is expected to be higher for anomalous samples than for benign samples.

4.2.4 Criteria-based De-Activation

To increase computational efficiency by decreasing the complexity of the ensembles of autoencoders without sacrificing accuracy, we propose to selectively de-activate some AEs and keep only $k' < N_{FS}$ number of AEs active in the ensemble.

In the criteria-based de-Activation approach, the AEs are activated or de-activated based on certain criteria (e.g., performance, energy constraints or requirements, etc.). In this way, this approach makes informed-decisions on de-activations. The criteria could be relaxed or constricted to scale up or down as required, thereby building an adaptive ensemble of AEs. In one embodiment of the proposed approach, the criteria to select AEs is based on training losses $L_{tr} = \{l_{tr}^1, ..., l_{tr}^{N_{FS}}\}$. Let $\mu_{L_{tr}}$, $\sigma_{L_{tr}}$ denote the mean and standard deviation of training losses of the ensemble. Then, any A_x whose training loss satisfies the criteria below is de-activated (either through post-training or in-training deactivation

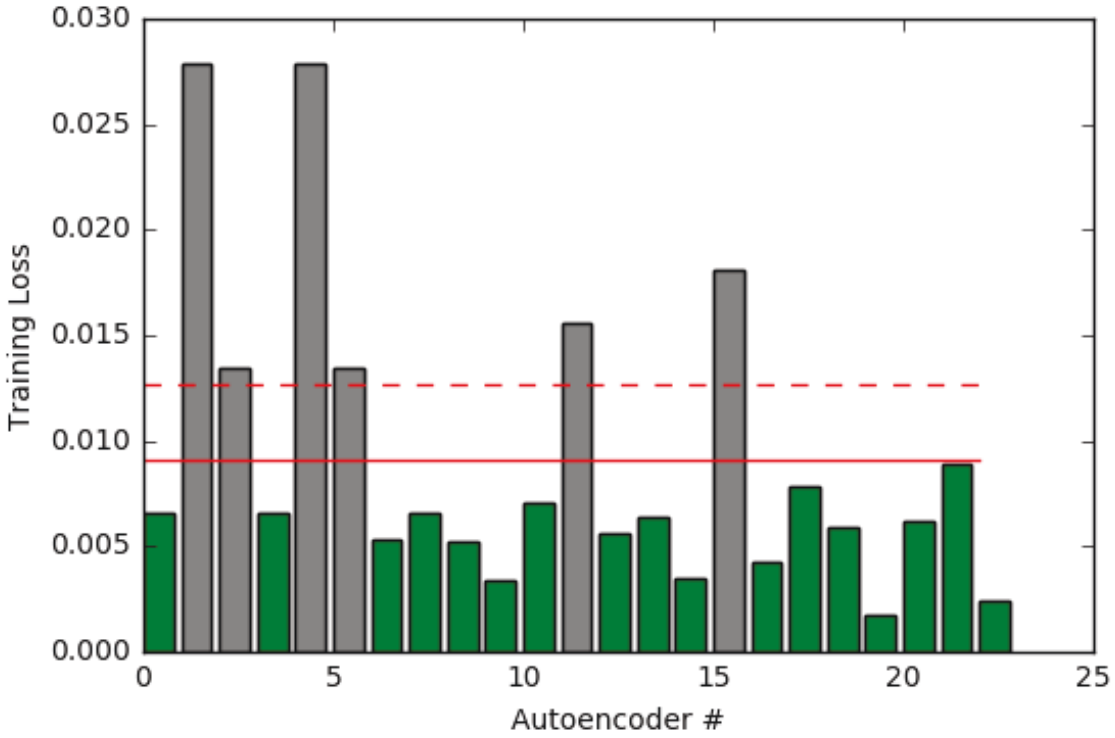


Figure 4.2: De-activating autoencoders based on Training Loss criteria, at $\beta = -0.5$. The red solid line represents the mean of training losses of all AEs in the ensemble whereas the red dotted line represents the threshold boundary as per criteria presented in Equation 4.2. The grey bars correspond to the training losses of AEs that are selected to be de-activated.

method):

$$l_{tr}^{x} > (\mu_{L_{tr}} - \beta * \sigma_{L_{tr})} \tag{4.2}$$

The result will be a set of $k' < N_{FS}$ active AEs in the ensemble. In Equation 4.2, β is a design parameter which can be decided by the user considering the level of computational saving required and tolerable reduction in accuracy (if any).

To illustrate the gains of the proposed crietria-based de-activations, in Figure 4.2,

we show the training losses of 23 autoencoders, trained for 23 feature subspaces on Kitsune [180] datasets (each subspace is of dimensionality $dim_{\mathbf{F_i}} = 5$). The grey bars depict the autoencoders whose losses satisfy Equation 4.2, whereas the green bars represent the autoencoders whose losses do not meet Equation 4.2. In this example, $\beta = -0.5$. After de-activations, only 17 out of 23 autoencoders would remain activated, yielding a computational saving of approximately 26.1%. Next, we describe how the criteria-based de-Activation approach is carried out with post- and in-training de-activation methods.

Criteria-based Post-Training De-activation (CPTD)

In the CPTD method, all the N_{FS} number of AEs in the ensemble layer are trained using the respective feature subspaces. Based on certain criteria (e.g., training loss, as presented in Equation 4.2), $k' < N_{FS}$ AEs are activated, reducing the number of AEs to be used by the output layer AE in the score aggregation module. While training the ensemble layers (without de-activations), we ignore the corresponding output layer's training, and use the active AEs only in making and training a narrower output layer. The scores (e.g. RMSEs) from each of the k' AEs are used to train the final output layer AE.

Criteria-based In-training De-activation (CITD)

In the CITD method, the AEs are de-activated during the training phase itself, possibly over a number of successive rounds defined by the parameter *N_DeAc_rounds*. The training data is split into *N_DeAc_rounds* number of batches, one for each round. We propose and evaluate the following approach to conduct criteria-based de-Activation-based in-training de-activation: Firstly, the ensemble layer AEs (represented by the set E) are trained over a certain number of rounds (*N_DeAc_rounds*), collecting their RMSEs in each round. At the end of each round, following the AE de-activation criteria Equation 4.2, certain AEs are de-activated. Also, the RMSEs corresponding to deactivated AEs are dropped. After the last round, one would have an ensemble E^f, and $RMSEs^f$ (the list of RMSEs for the final subset of active AEs from all rounds' training).

In the final output layer, an AE is built with the following configurations: the number of input and output units is equal to the number of active AEs in E^f. The output layer

Table 4.2: Description of Kitsune Datasets [180]: Types of Attacks and Number of Samples (Benign and Malicious)

Attack Category	**Attack**	**Description**	**#Ben**	**#Mal**
Botnet Malware	Mirai	IoT device credentials exploited to inject Mirai malware and hunt for new victims	121,621	642,516
Reconnaissance	OS Scan	Hunts for vulnerabilities in hosts and operating systems in the network	1,632,151	65,700
	Fuzzing	Random commands used to hunts for vulnerabilities in cameras' web servers	1,811,356	432,783
Man in the middle	Video Injection	Other/recorded videos injected into the live video streams	2,369,902	102,499
	ARP MitM	ARP Poisoning exploited to intercept LAN traffic	1,358,995	1,145,272
	Active Wiretap	Exposed cables exploited to setup a covert network bridge (wiretap) to intercept LAN traffic	1,355,473	923,216
Denial of Service	SSDP Flood	Cameras exploited to generate UPnP advertisements to overload/spam the recording server(s)	2,637,662	1,439,604
	SYN DoS	A camera's web server is overloaded to disable the camera's video stream	2,764,238	7,038
	SSL Renego-tiation	A camera is overloaded with SSL renegotiation packets to disable its video stream	2,114,919	92,652

Table 4.3: Attack Types in NBaIoT Dataset [175]

Attack Type	**Botnet Family**	**Description**
Scan	Mirai and BASHLITE (Gafgyt)	Looks for vulnerable devices in the network
Junk	BASHLITE (Gafgyt)	Sends junk/spam data
COMBO	BASHLITE (Gafgyt)	Sends spam data; Opens connection to a given IP address and port
Flooding	Mirai	ACK, SYN, UDP, UDPplain (higher PPS, enabled by fewer options)
	BASHLITE (Gafgyt)	UDP, TCP

Table 4.4: Composition of NBaIoT Datasets [175]: IoT devices, and number of benign and malicious samples

ID	**Device**	**#Benign**	**#Mirai**	**#Gafgyt**
1	Danmini (Doorbell)	40395	652100	316650
2	Ecobee (Thermostat)	13111	512133	310630
3	Ennio (Doorbell)	34692	N/A	316400
4	Philips B120N10 (Baby Monitor)	160137	610714	312723
5	Provision PT737E (Security Camera)	55169	436010	330096
6	Provision PT838 (Security Camera)	91555	429337	309040
7	Samsung SNH1011N (Webcam)	46817	N/A	323072
8	SimpleHome XC57-1002-WHT (Security Camera)	42784	513248	303223
9	SimpleHome XC57-1003-WHT (Security Camera)	17936	514860	316438

AE (A_{OP}) is trained using the stored RMSEs (from all training rounds) of the active AEs in E^f.

In each round of ensemble layer's training, we restrict the de-activations such that the ensemble size doesn't go below a minimum, defined by the parameter *Min_E_Size*. In any given round, de-activations will not be performed if the resulting ensemble would be smaller than *Min_E_Size* and the training will proceed to the next round using the same set of AEs as in the current round. With such a restriction, one may comprehend that the de-activations could result in an ensemble of size *Min_E_Size* much earlier than *N_DeAc_Rounds*. In such cases, we do not stop the training but rather complete all training rounds to learn over the entire set of training samples. Stopping early may result in an under-trained ensemble. Another way of carrying out the de-activations is by having an additional restriction which limits the number of de-activations in each training round. This would have the advantage of preventing early shrinking of the ensemble. In Section 4.4, we study and evaluate both of the above mentioned ways of restricting in-training de-activations in criteria-based de-Activation.

4.2.5 Random De-Activation

Since larger ensembles require more time for training and consume more computational resources in inference than smaller ensembles, we propose random de-activation of some AEs. Although this idea of ours may appear similar to the Dropout method [243], there is a striking dissimilarity. While the dropout method of [243] carries out dropping of certain parts of the hidden units in selected layers of a neural network during training, and the dropout method of [140] drops out some layers of a deep network during training, our random de-activation method drops an entire AE (deep or shallow) in an ensemble during training and/or testing.

The difference between criteria-based (described in Section 4.2.4) and random de-activation is that the latter does not take into account factors such as an individual AE's performance. Next, we describe how the random post-training and in-training de-activation methods are carried out.

Random Post-Training De-activation (RPTD)

The RPTD method is implemented as follows. First, the ensemble layers are trained without any de-activations. Second, $k' < N_{FS}$ AEs are randomly activated, keeping the remaining $N_{FS} - k'$ de-activated. Each AE (represented by A_i) in E has an equal probability of being de-activated, i.e., no AE has priority over another AE in being active. The number of AEs to de-activate depend on the $DeAc\_Ratio$ parameter. For example, a $DeAc\_Ratio = 0.1$ implies de-activating 10% of the AEs in the ensemble (rounded using floor function). The output layer AE in the score aggregation module (A_{OP}) is then trained using the scores (e.g., RMSEs) of only the k' active AEs.

Random In-Training De-activation (RITD)

The RITD method is implemented as follows. The training process of the ensemble layers is carried out over a number of rounds given by the parameter N_DeAc_rounds. The training data is split into N_DeAc_rounds number of batches such that each round works on its batch of data. In each round, the number of AEs to be de-activated (N_{drop}) depends on the $DeAc\_Ratio$ parameter. The de-activations are controlled by the parameter Min_E_Size to maintain a minimum ensemble size. Let C_d^r be the set of AEs randomly selected to be de-activated in a certain round r (where the size of C_d^r is given by $|C_d^r| = DeAc\_Ratio \cdot |E_r|$). If it is the case that the ensemble size $|E^r|$ in a given round r, after de-activating $|C_d^r|$ number of the AEs from the ensemble, would go below Min_E_Size, the number of AEs to de-activate is determined by $N_{drop}^r = |E^r| - Min\_E\_Size$.

After the final round f, we would have the ensemble E^f, and the scores $RMSEs^f$ (i.e., the RMSEs scores for the AEs in E^f from all rounds of training). The output layer AE (A_{OP}) in the score aggregation module is then trained based on $RMSEs^f$. The A_{OP} would have $|E^f|$ number of input and output units.

4.3 Experimental Setup

In this section, we describe the realistic IoT intrusion attacks datasets used in this chapter to evaluate the proposed methods. In addition, we define the performance metrics which shall be used to evaluate and compare the proposed methods.

4.3.1 Datasets

In this work, we evaluate the proposed methods using two groups of recent realistic IoT datasets: (i) *Kitsune* [180] and (ii) *NBaIoT* datasets [175] .

The Kitsune datasets were collected from real IoT devices such as IP cameras in a surveillance network deployment across two connected sites. Table 4.2 outlines the different attack types and benign/malicious samples for each. For each dataset, we consider the first 1 million packets (benign only) to train the ensemble of AEs in an unsupervised fashion while the rest are split equally for validation and testing. In the case of Mirai dataset though (due to lesser number of samples than other datasets), we use only 50% of benign data in training, 25% for validation and 25% for testing. Since the goal of this chapter is on developing an unsupervised learning-based intrusion detection method, the malicious samples are not used in training the ensembles.

The NBaIoT datasets of [175] were collected from a real testbed of nine wireless-ly connected IoT devices, setup in a way to resemble an enterprise setting. These devices were infected with two families of real-world IoT-based botnets (Mirai and BASHLITE/Gafgyt). The datasets contain packet- and flow-based features representing the benign and malicious traffic.

Table 4.3 provides a summary of the different attacks covered in the NBaIoT datasets under the Mirai and BASHLITE botnet families. The composition of the dataset is given in Table 4.4. We split the benign samples of the datasets into training, validation and testing sets using 50%-25%-25% ratio. The benign traffic data for each IoT device consists of frequent and infrequent actions as well. The malicious samples are utilized only for validation and testing.

4.3.2 Performance Metrics

The performance metrics we use to assess the effectiveness of the proposed methods include the following, which are based on True Positives (TPs), True Negatives (TNs), False Positives (FPs) and False Negatives (FNs). For some input sample, given A_{OP}'s output which is an anomaly score based on reconstruction errors, a threshold Φ is applied to classify the sample as normal/anomalous. Φ could be adjusted to enhance or relax the classification effectiveness.

- *Area under the Curve (AUC)*: The area under a receiver operating characteristics (ROC) curve. It gives the probability of a randomly chosen anomalous sample being ranked by a classifier higher than a randomly chosen normal sample. The higher the AUC, the better the algorithm or classifier.
- *Equal Error Rate (EER)*: The measure of FP rate (FPR) and FN rate (FNR) when these are minimal and equal, across all possible values of Φ. Lower EER indicates a better classifier.
- *True Positive Rate (TPR)*: The ratio between TPs and sum of TPs and FNs. We measure the TPR at a Φ that yields a low FPR, e.g. 0.001.

4.4 Results and Discussions

We investigate the performance of the proposed Criteria-based and Random, Post-Training and In-Training de-activation methods. To keep the complexity low while achieving high accuracies, we employ AEs of one hidden layer each. The input and output layers are of size 5 neurons each. In other words, $dim_{\mathbf{F_i}} = 5$ for each $\mathbf{F_i}$, and so the total number of AEs and feature subspaces, $N_{FS} = 23$. The performance of the proposed methods is compared against that of other methods on the Kitsune [180] and NBaIoT datasets [175], to highlight the advantages and gains of the proposed methods.

In the context of our work which is based on unsupervised learning, the Equal Error Rate (EER) metric has been used to evaluate how good a method is in lowering False

Positive and False Negative rates. Lower the EER, better the method is in differentiating malicious anomalies from benign samples. It is also worth highlighting that in a production environment, the proposed methods could be efficiently deployed as lightweight tools to raise alarms to the network operators based on the detected anomalies. More sophisticated automated or manual methods may be employed to verify if the detected anomalies are malicious or otherwise. Achieving a fully automated intrusion detection system that yields zero false positive and zero false negative rates is an ongoing pursuit.

4.4.1 Performance Evaluation of Criteria-based De-Activations

In this section, we evaluate the criteria-based post- and in-training methods, referred to as *CPTD* and *CITD* respectively. In addition, we examine the comparisons between CPTD, CITD and the case of no de-activations.

Evaluation of CPTD

In Figure 4.3 we show how AUC and number of active AEs vary with respect to the parameter β of Equation 4.2, using the Mirai dataset of [180]. As one can observe, increasing β reduces the number of active AEs in the ensemble which also impacts the AUC metric, reducing the classification accuracy.

Lower levels of Beta, e.g., $\beta = -0.5$, yield AUCs of around 0.9454 with 17 active AEs in the ensemble after de-activation. One could achieve AUCs of above 0.9 with an ensemble reduced to as low as 5 AEs (e.g., Figure 4.3, $\beta = 0.6$).

Evaluation of CITD with Minimum Ensemble Size (*Min_E_Size*) Restriction

In Figure 4.4, we show the number of active AEs per round, given *Min_E_Sizes* of $3, 5, 7, 10$, and varying β from -0.5 to 0.5 in steps of 0.1. In general, one can observe that a higher β leads to a higher de-activation rate in the initial rounds of training. When one observes the AUC vs β curves in Figure 4.5, it seems that a lower β (and hence lower de-activation rate in initial rounds) yields higher AUCs. At $\beta = 0$ however, one may note an increased

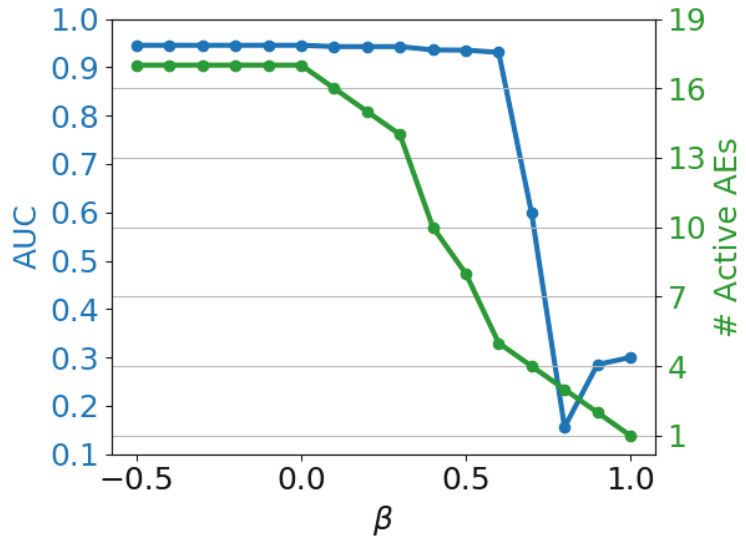


Figure 4.3: Criteria-based Post-Training De-activation: AUC and Number of Active AEs (*#ActiveAEs*) with respect to varying β (Equation 4.2). Initial ensemble size (prior to deactivations) is 23.

AUC. For example, with *Min_E_Size*=10, AUC at $\beta = 0$ is higher than at $\beta = -0.5$. This could be understood by looking at Figure 4.4 which shows that at $\beta = -0.5$, *#ActiveAEs* were down to 10 in round 3, whereas at $\beta = 0.0$, *#ActiveAEs* were down to 10 in round 4, indicating that a higher number of initially held active AEs at $\beta = 0.0$ may have lead to a more accurate ensemble. Based on Figure 4.5, we use $\beta = -0.5$ in our experiments in Section 4.4.1.

One can observe from Figure 4.4 that a deactivation restriction based on *Min_E_Size* alone could lead to cases of all or none de-activations of candidate AEs (in a round). In such cases, the AEs are deactivated to a certain level which is above the *Min_E_Size* and then the method stops deactivating the set of candidate AEs so as not to violate the restriction. For example, in Figure 4.4(b), when $\beta = 0.5$, although many AEs get deactivated in round 1, there are no further de-activations until round 9 in which the ensemble has 5 active AEs. This observation motivates another approach (see next subsection) to restrict the de-activations in a way that ensures some de-activations are performed in every round, by adding the flexibility to de-activate only a certain percentage of candidate AEs in cases

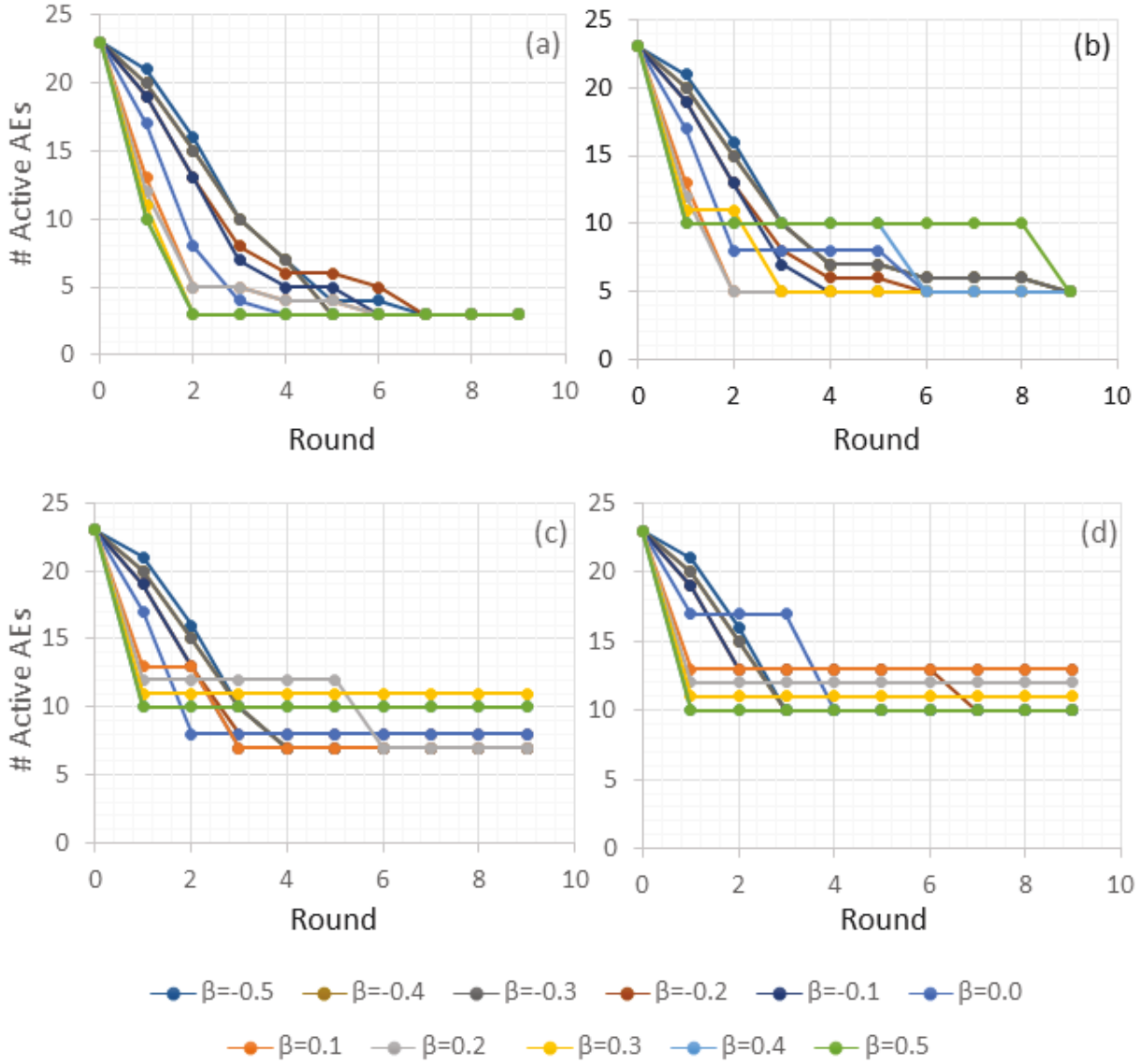


Figure 4.4: CITD with Minimum Ensemble Size (*Min_E_Size*) Restriction-based deactivations: Number of active AEs per round, for varying β (-0.5 to 0.5), and Min_E_Size = (a) 3, (b) 5, (c) 7, and (d) 10.

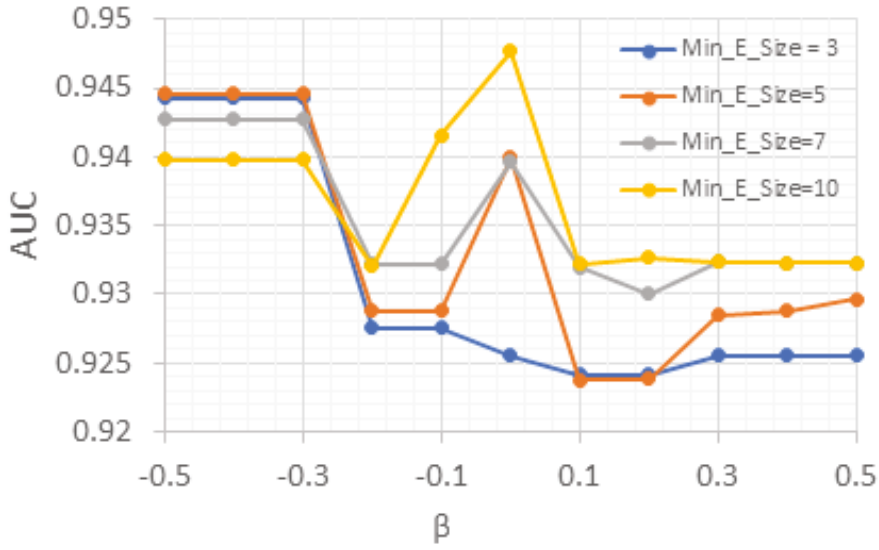


Figure 4.5: CITD: AUC vs β for different Min_E_Size, with Minimum Ensemble Size restriction-based deactivations.

where de-activating all candidate AEs could make the ensemble smaller than *Min_E_Size*.

Evaluation of CITD with Maximum Number of De-Activations Per Round (*Max_DeAc*) Restriction

In the previously discussed minimum ensemble size-based restriction, we noted that in each round, the method finds a candidate set of AEs to de-activate. However, it will only de-activate these AEs if the ensemble size doesn't go below *Min_E_Sizes*. It had a limitation of not being able to selectively de-activate some (if not all) of these AEs. In this subsection, we investigate adding a restriction based on the maximum number of de-activations per round. Consider for example, in round r, given a set E^r of AEs in the ensemble. The method finds a candidate set of AEs to de-activate, C_d^r. From C_d^r, in round r, the number of AEs to drop are determined by

$$N_{drop}^r = min(|C_d^r|, Max\_DeAc) \tag{4.3}$$

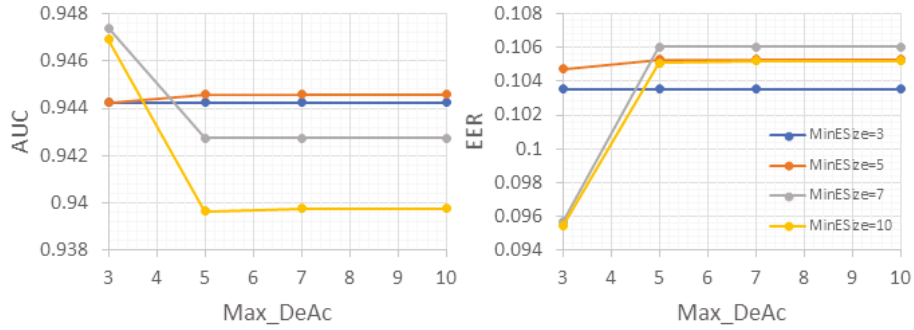


Figure 4.6: Evaluation of CITD with Maximum Number of De-Activations Per Round (*Max_DeAc*) Restriction: AUC and EER vs *Max_DeAc* for different *Min_E_Size*.

where *Max_DeAc* is a design parameter to restrict the maximum number of AEs that may be deactivated in any given round. So, only N^r_{drop} number of AEs with highest training losses (l_{tr}) in C^r_d are de-activated. Such a restriction is expected to avoid a high de-activation rate in the initial rounds, leading to more accurate ensemble, as observed previously.

In Figure 4.6, we present the results of investigating the additional restriction of maximum number of de-activations per round. For these experiments, we fix $\beta = -0.5$ and study the performance of ensembles with *Min_E_Size=3,5,7,10* and *Max_DeAc=3,5,7,10*, over *N_DeAc_Rounds=10* rounds. Observing Figure 4.6, it is clear that AUC and EER are mainly affected by the final ensemble size and that *Max_DeAc=3* and *Min_E_Size=7* suits best (in terms of higher AUC and lower EER)

From the figures showing the *#ActiveAEs* per round for different ensemble sizes (Figure 4.7), one can note that an increase in *Max_DeAc* seems to be increasing the rate of de-activations, especially in the initial rounds. In later rounds, when #ActiveAEs goes below *Max_DeAc*, #ActiveAEs to be dropped then relies on the *Min_E_Size* restriction only. In these graphs, #ActiveAEs per round with *Max_DeAc* set to 7 and 10 follows the same trend regardless of the ensemble size. This indicates that at higher *Max_DeAc*, the AEs dropped in each round is limited mainly by the number of candidate AEs in C^r_d (which tend to be lower than these *Max_DeAc*'s), as per the equation of N^r_{drop}.

Table 4.5: Performance Comparison of CPTD and CITD Methods vs No de-activations

Attack	ARP MitM				Fuzzing			
Method / Metric	AUC	TPR	EER	#AEs	AUC	TPR	EER	#AEs
No De-Ac	0.86	0.06	0.187	23	0.296	0.024	0.636	23
CPTD	0.88	0.029	0.204	17	0.342	0.024	0.613	17
CITD	0.89	0.031	0.179	7	0.336	0.024	0.642	7
Attack	Mirai				OS Scan			
Method / Metric	AUC	TPR	EER	#AEs	AUC	TPR	EER	#AEs
No De-Ac	0.943	0.878	0.105	23	1.0	1.0	0.0	23
CPTD	0.945	0.887	0.096	17	0.999	1.0	3.16e-5	17
CITD	0.947	0.887	0.096	7	0.999	1.0	2.1e-5	7
Attack	SSDP Flooding				SSL Renegotiation			
Method / Metric	AUC	TPR	EER	#AEs	AUC	TPR	EER	#AEs
No De-Ac	0.999	0.999	0.0005	23	0.980	0.167	0.067	23
CPTD	0.999	0.999	0.0006	16	0.967	0.168	0.108	16
CITD	0.999	0.999	0.0005	8	0.956	0.092	0.125	8
Attack	SYN DoS				Video Inj.			
Method / Metric	AUC	TPR	EER	#AEs	AUC	TPR	EER	#AEs
No De-Ac	0.976	0.0	0.062	23	0.89	0.0091	0.1788	23
CPTD	0.896	0.0	0.119	18	0.878	0.008	0.175	21
CITD	0.52	0.0	0.478	7	0.611	0.0001	0.459	9
Attack	Wiretap							
Method / Metric	AUC	TPR	EER	#AEs				
No De-Ac	0.989	0.801	0.045	23				
CPTD	0.985	0.3696	0.061	21				
CITD	0.842	0.364	0.222	7				

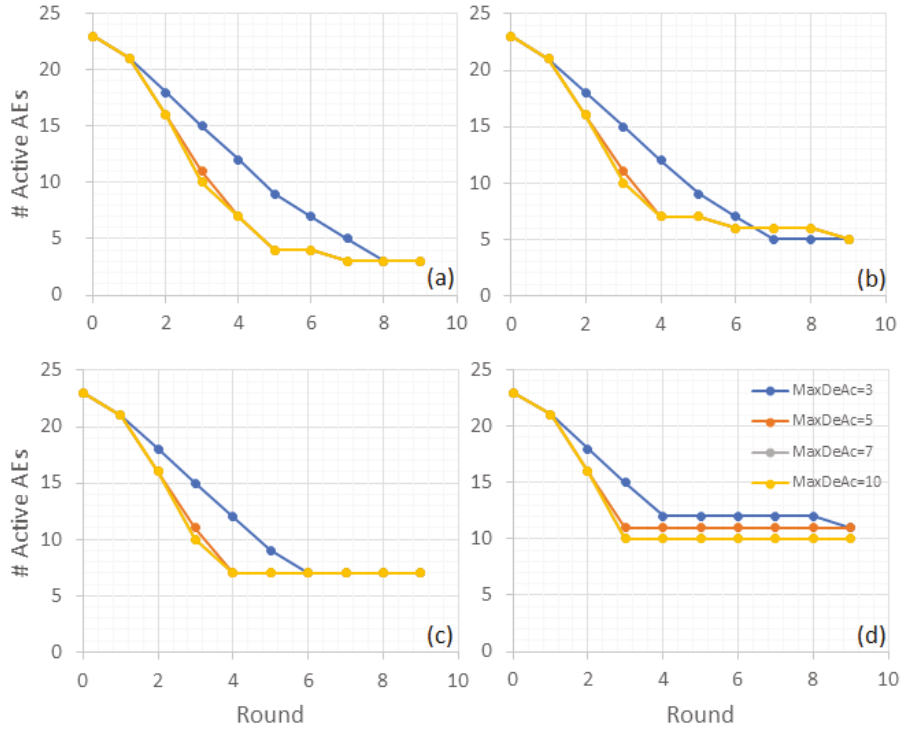


Figure 4.7: CITD: Number of active AEs per round, with $\beta = -0.5$ and Min_E_Size = (a) 3, (b) 5, (c) 7, and (d) 10, with the additional restriction based on maximum number of deactivations per round.

Detection of Intrusion Attacks: Comparison of CPTD, CITD vs No De-activations

In Figure 4.8 (and Table 4.5), we summarise the performance evaluation (in terms of AUC, TPR at FPR= 0.001, and EER) of CPTD and CITD methods to compare against the baseline ensemble that is similar to Kitsune [180] and does not have any de-activations. For these experiments, the following parameters were used: PTD ($\beta = -0.5$), ITD ($\beta = -0.5$, *N_DeAc_rounds* = 10, *MaxDeAc=3*, *Min_E_Size=7)*. As one can observe, both PTD and ITD methods yield ensembles of lower size (and hence lower complexity), measured in terms of the number of AEs active in the ensemble (see #AEs in Table 4.5).

The ensemble size serves as a good metric to assess the inference time-cost and computational load instead of measuring run-time or memory usage because devices may differ in

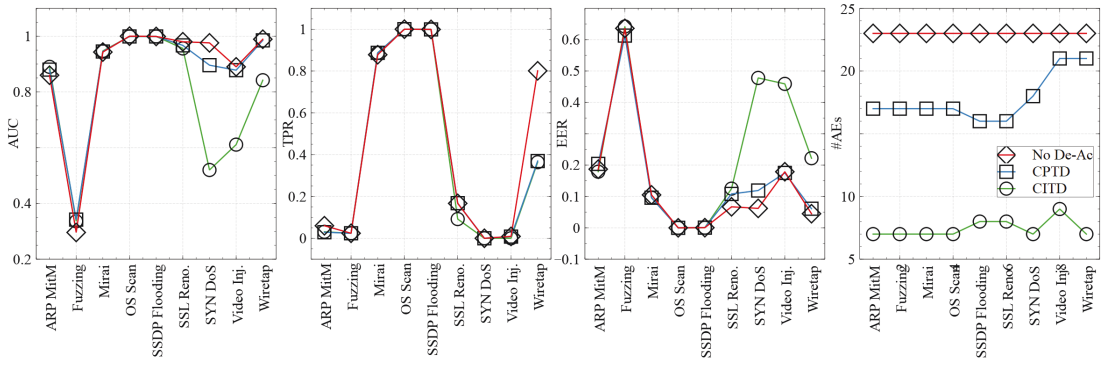


Figure 4.8: Performance Comparison of CPTD and CITD Methods vs No de-activations

their computational power, memory efficiencies, hardware accelerations, etc. Regardless of these differences, the smaller an ensemble's size, the lesser its computational and memory load would be. The CITD method achieves a larger reduction in the ensemble size in comparison to that of CPTD. A lower ensemble size (i.e., lesser number of active AEs) translates to faster inference time because fewer AEs would need to be executed. Hence, both CPTD and CITD would incur lesser inference time costs than an ensemble without any de-activations. Moreover, an ensemble with fewer number of autoencoders would require lesser RAM/CPU usage than an ensemble with more number of autoencoders.

Both methods achieve AUC, TPR and EER scores better or as closely good to those with the baseline ensemble, on most attack datasets. The CPTD and CITD methods lowered the AUC for some datasets, e.g., SSL Renegotiation, SYN DoS, and Video Injection (see Table 4.5). One possible reason is that since CITD and CPTD do not take into account feature importance, it is highly possible that they deactivate AEs corresponding to feature subspaces that are important for such types of attacks. In a future work, we plan to incorporate feature importance into the CITD and CPTD methods to learn more robust ensembles. Nonetheless, the above results validate the effectiveness of the proposed methods in reducing ensemble complexity while attaining good detection results on most attacks. In massive IoT environments where an IDS is composed of ensembles of AEs for each device or device-type, our methods provide for more efficient ensembles which could be scaled up or down adaptively.

4.4.2 Performance Evaluation of Random De-Activations

In this section, we evaluate the random post- and in-training de-activation methods, referred to as *RPTD* and *RITD* respectively. We also examine the comparisons between RPTD, RITD and the case of no de-activations.

Random Post-Training De-activations (RPTD)

In the RPTD method, the *DeAc_Ratio* parameter decides the number of active AEs that remain in the ensemble after random de-activations. We study the performance of RPTD under different *DeAc_Ratios* ranging from 0.1 to 0.9. We show in Figure 4.9, the AUC scores of RPTD method for each of the nine attacks datasets.

For the ARP MitM, SSL Renogiation, SYN DoS, Video Injection and Wiretap attacks datasets, we observe that a higher *DeAc_Ratio* generally reduces the AUC scores. This is expected as higher the number of de-activated AEs, higher the number of de-activated corresponding feature subspaces would be, which could lead to the case of de-activating feature subspaces that are important to identify these kinds of attacks. An interesting observation was made with regards to the Fuzzing attacks dataset. A slightly better performance was achieved with higher *DeAc_Ratio*. This could be due to de-activations of AEs and feature subspaces that were redundant or noisy for the purposes of identifying Fuzzing attacks. Nonetheless, the overall AUC scores for Fuzzing attacks dataset were lower than 0.5.

However, for the Mirai, OS Scan and SSDP Flooding attacks datasets, we find that the *DeAc_Ratio* does not have a major impact on the AUC scores even for *DeAc_Ratio* as high as 0.9. This indicates that even a handful of AEs (and corresponding feature subspaces) are sufficient to successfully identify such kinds of attacks.

In addition to the above observations, one may note some fluctuations in AUCs as *DeAc_Ratio* increases. This is expected since the de-activations are random (with no consideration of the individual AEs' performance), it may so happen that important AEs are de-activated. Similar observations are made when we look at EERs of the RPTD method on the Kitsune datasets, shown in Figure 4.10.

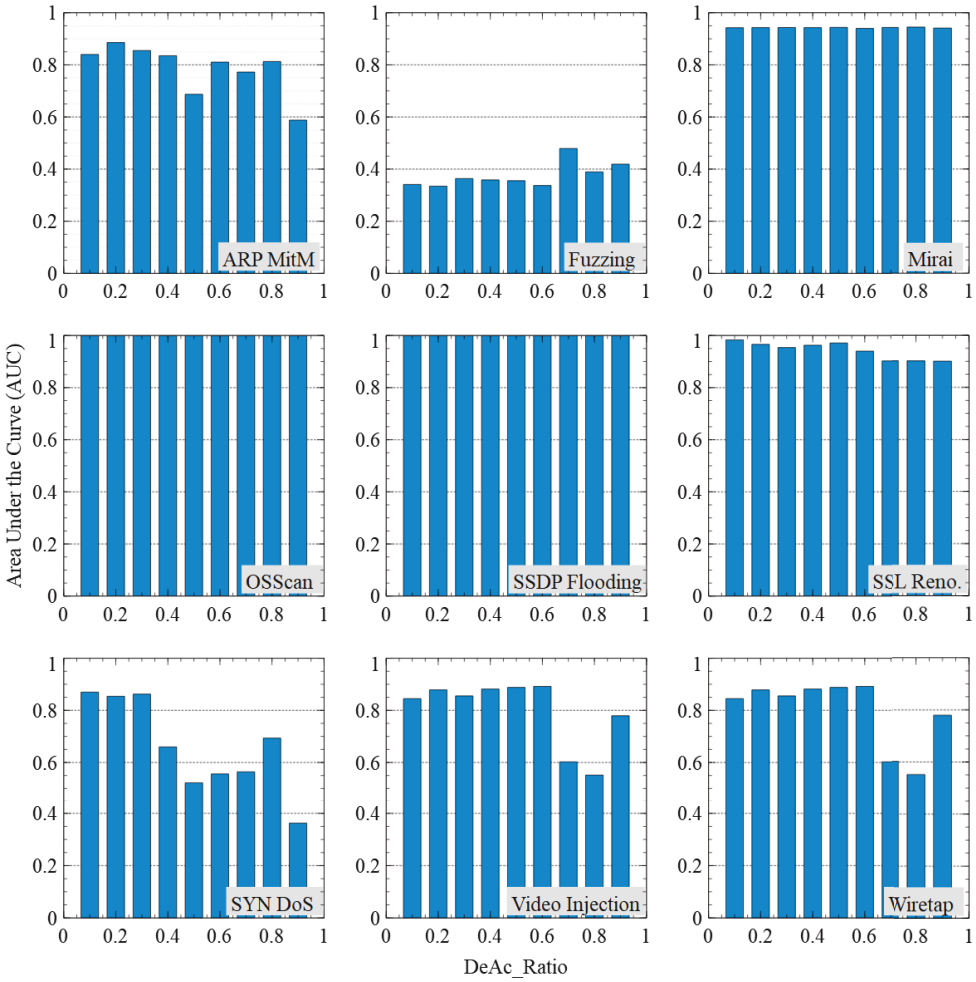


Figure 4.9: Evaluating RPTD for the nine attacks datasets of Kitsune [180]: Showing how AUC varies with *DeAc_Ratio* (Horizontal axes represent the *DeAc_Ratio* while the vertical axes depict the AUC scores.

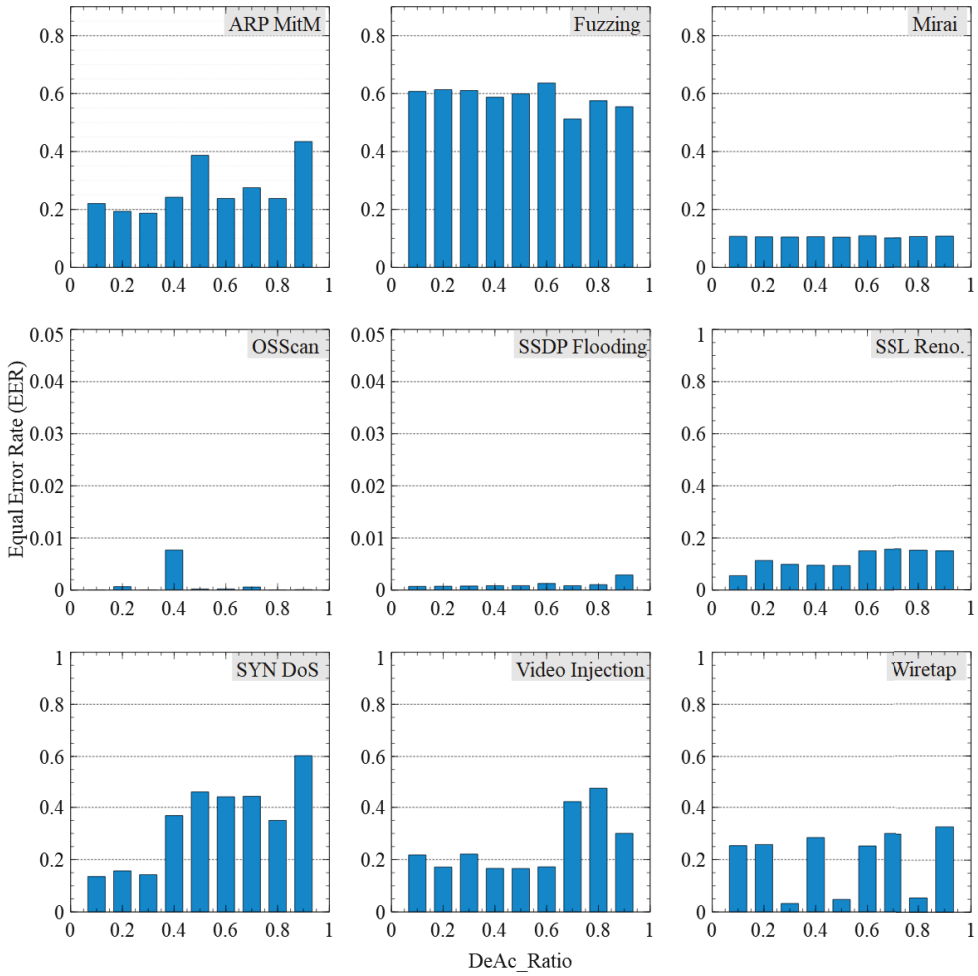


Figure 4.10: Evaluating RPTD for the nine attacks datasets of Kitsune [180]: Showing how EER varies with *DeAc_Ratio* (Horizontal axes represent the *DeAc_Ratio* while the vertical axes depict the EER scores.

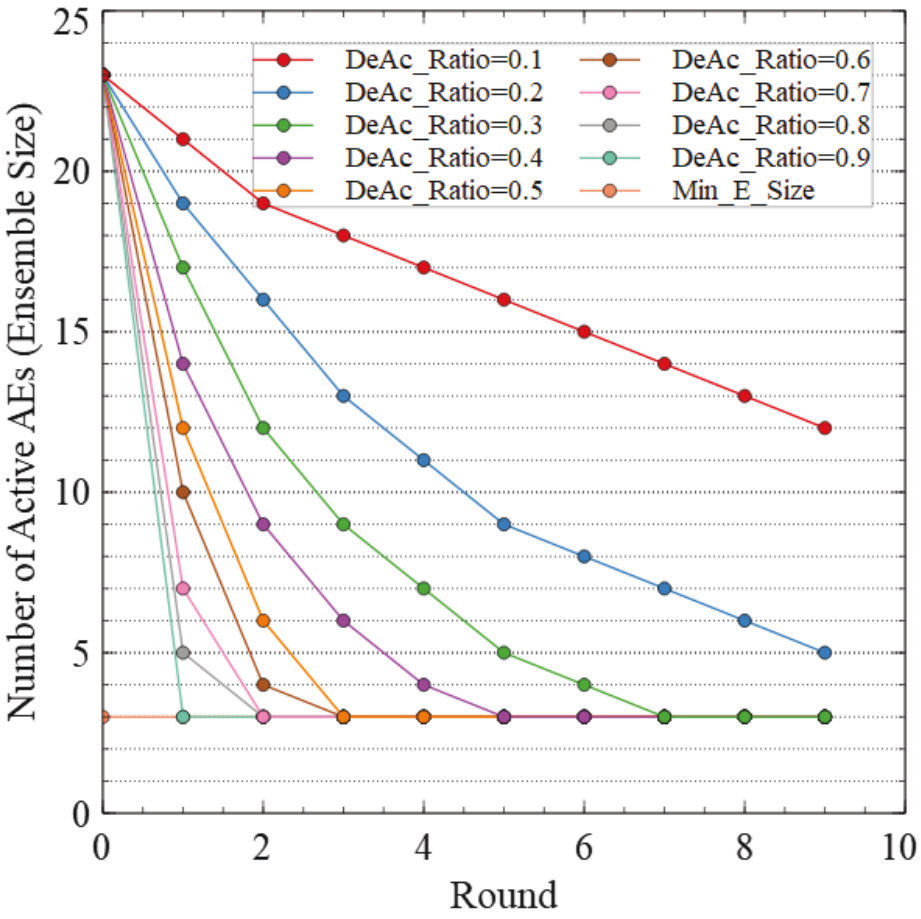


Figure 4.11: RITD: Number of active AEs per round for different *DeAc_Ratio* per round and $Min\_E\_Size = 3$.

Random In-Training De-activations (RITD)

In the RITD method, the random de-activations are performed during training over a number of rounds (i.e., N_DeAc_rounds). In each round a $DeAc\_Ratio$ determines the number of AEs to de-activate, restricted by a minimum ensemble size Min_E_Size. In Figure 4.11, we show how number of active AEs per round vary for different $DeAc\_Ratios$. In these experiments, the training was conducted over $N\_DeAc\_rounds = 10$ and $Min\_E\_Size = 3$. As one can observe in Figure 4.11, a higher $DeAc\_Ratio$ per round leads to earlier reduction in ensemble size.

Next, we would like to see if $DeAc\_Ratio$ has any influence on the performance of RITD method in terms of AUC and EER scores. While Figure 4.12 shows the AUC scores for the nine attack datasets, Figure 4.13 shows the EER scores. One can see that for the Mirai, OS Scan, SSDP Flooding and SSL Renegotiation, the performance was good across different $DeAc\_Ratios$ (high AUCs and low EERs).

In the case of ARP MitM, Fuzzing, SYN DoS, Video Injection, and Wiretap attacks, the performance seemed to fluctuate with changing $DeAc\_Ratios$, not showing any consistent increase (or decrease) with an increase in $DeAc\_Ratio$. We attribute this to the random de-activation approach that does not take into account the value or importance of AEs. So, it is possible that despite a lower $DeAc\_Ratio$, the method randomly selected important AEs to de-activate in each round and hence the overall performance in terms of AUC would go low and increase the EER. On the other hand, it is possible that despite a higher $DeAc\_Ratio$, the method randomly de-activates un-important or redundant AEs and hence the AUC increases and EER decreases.

Comparison of RPTD, RITD and No De-activations

In Figures 4.14 and 4.15, we present the AUC, EER and TPR scores as well as final ensemble size ($\#AEs$) for the RPTD and RITD methods.

Here, we compare the two methods based on the performance of the medium and small ensembles, i.e., of ensembles that are around 50% and 90% smaller than the original ensemble. With RPTD, medium ensembles (of size 12) are achieved with $DeAc\_Ratio = 0.5$

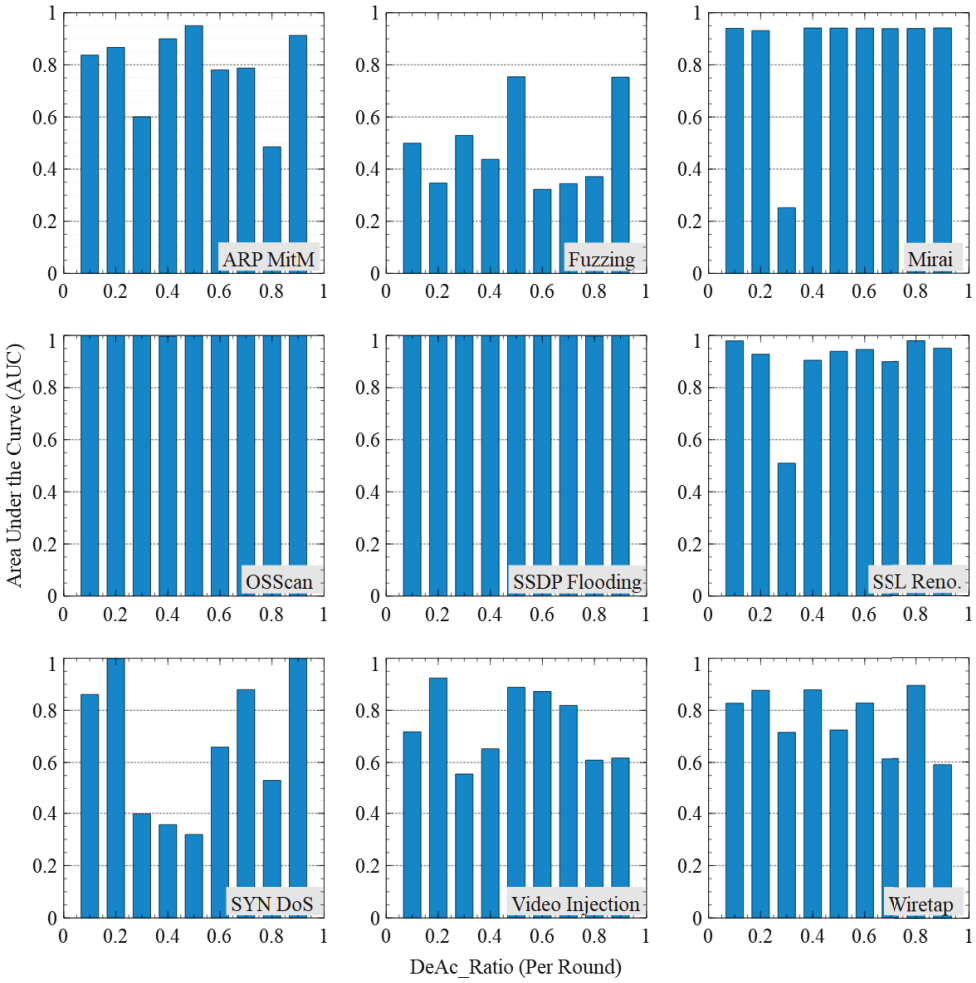


Figure 4.12: Evaluating RITD for the nine attacks datasets of Kitsune [180]: Showing AUC scores with different *DeAc_Ratio* per round, for $N\_DeA\_Rounds = 10$.

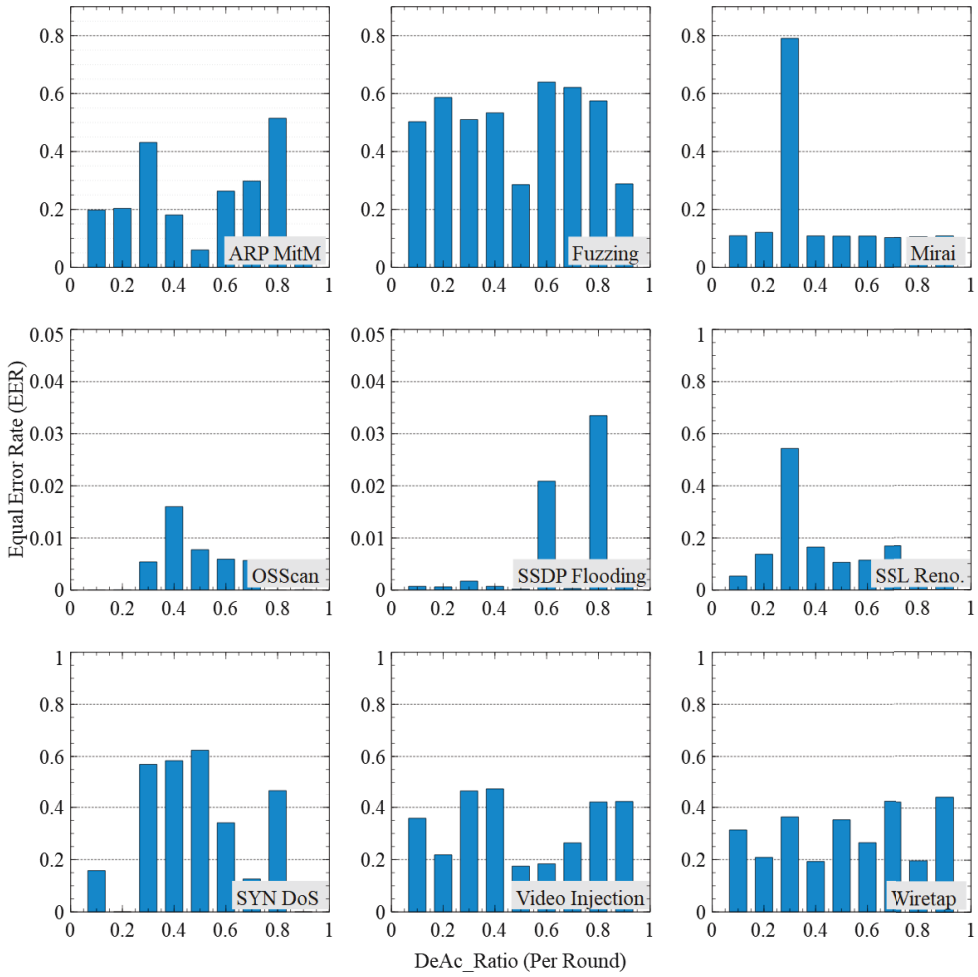


Figure 4.13: Evaluating RITD for the nine attacks datasets of Kitsune [180]: Showing EER scores with different *DeAc_Ratio* per round, for $N\_DeA\_Rounds = 10$.

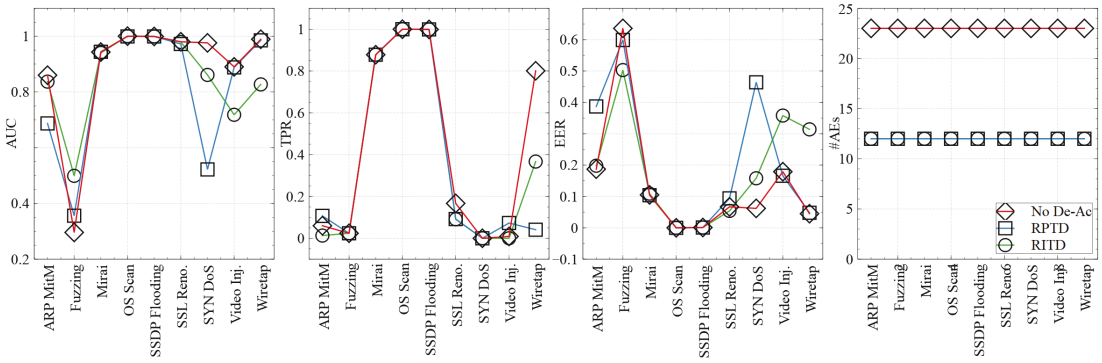


Figure 4.14: Performance Comparison of Medium-sized Ensembles (RPTD and RITD methods)

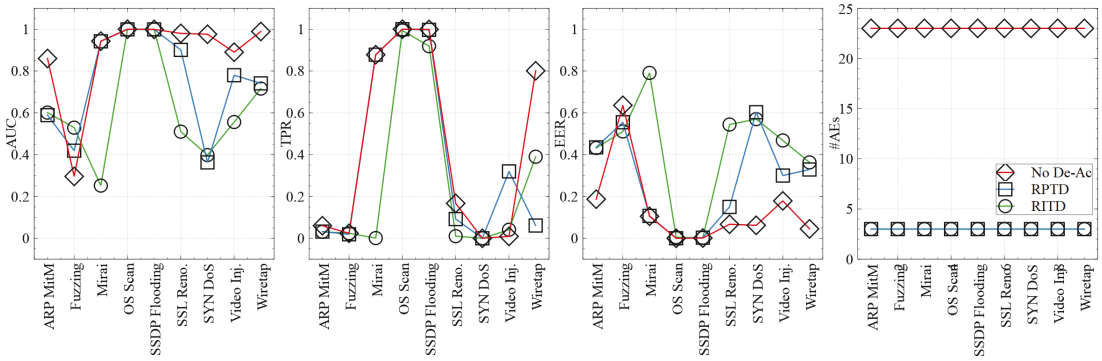


Figure 4.15: Performance Comparison of Small-sized Ensembles (RPTD and RITD methods)

whereas small ensembles (of size = $Min\_E\_Size = 3$) are achieved with $DeAc\_Ratio = 0.9$. As for RITD, medium ensembles are achieved with $DeAc\_Ratio = 0.1$ (per round) whereas small ensembles are achieved with $DeAc\_Ratio$ of atleast 0.3 (per round).

Table 4.6 presents the performance scores (AUC, TPR at FPR=0.001, EER) of medium-sized ensembles learnt through RPTD and RITD methods. It also compares against the scores achieved by the original ensemble without de-activations. Comparing between RPTD and RITD, for most attacks, RITD was found to perform better than RPTD with higher AUC and lower EER scores. The only exceptions to this were the Video Injection and Active Wiretap attacks. A possible reason could be that, for these attacks datasets, the RITD method randomly de-activated important AEs early on and hence the ensemble may have been left with weaker AEs.

Looking at the performance of small ensembles learnt through RPTD and RITD methods (see Table 4.7), we find similar observations as with medium-sized ensembles. RITD was found to perform better than or similar to RPTD for ARP MitM, Fuzzing, OS Scan, SSDP Flooding, and SYN DoS attacks datasets. However, for Mirai, SSL Renegotiation, Video Injection and Active Wiretap attacks, RPTD performed better. The EERs for the RITD method were higher than for the RPTD method, indicating that the AEs in the final ensemble were weaker as important AEs may have been de-activated through random selection.

In Tables 4.6 and 4.7, comparing the scores of RPTD and RITD methods against the scores of ensembles with no de-activations, we find that the performance of small ensembles suffers more than that of medium ensembles. Although smaller ensembles may achieve lower accuracy related performance, these require lesser computation and storage resources than larger ensembles. The smaller ensembles could thus be deployed as lightweight triggers that activate, when needed, larger ensembles for more accurate detections.

4.4.3 Comparison with Other Methods (on Kitsune Datasets)

Comparing AUCs

We compare the performance of CPTD, CITD, RPTD, and RITD methods against the performance of the baseline method of no de-activations and that of other methods reported in [180]. Table 4.8 presents the AUC scores.

The scores of the proposed methods (CPTD, CITD, RPTD, RITD) are based on final ensembles of sizes = $Min\_E\_Size = 7$, i.e., less than a third of the size of the original ensemble with no de-activations. The $DeAc\_Ratio$ for the RPTD method was 0.7 whereas for the RITD method, $DeAc\_Ratio$ per round was 0.2. The scores of Kitsune, Isolation Forest and Gaussian Mixture Model (GMM) methods are as reported in [180] for each attack dataset.

In terms of AUC, the proposed methods achieved highest scores on the ARP MitM, OS Scan, SSDP Flooding, and SSL Renegotiation attacks datasets. In the case of SSDP Flooding attacks, the proposed methods and the other methods performed very well ($AUC = 0.999$), the only exception being the Isolation Forest method which yielded an $AUC = 0.948$. As for Mirai, Video Injection and Wiretap attacks datasets, the proposed methods achieved second highest scores. These scores demonstrate the effectiveness of the proposed methods with much smaller ensembles of AEs.

Within the suite of proposed methods, we find that criteria-based de-activations outperformed random de-activations on ARP MitM, Mirai, SYN DoS datasets and Wiretap attacks datasets and achieved similar performance as random de-activations on OS Scan and SSDP Flood datasets. RITD showed better AUC in the face of Fuzzing, OS Scan, and SSL Renegotiation attacks. As for Video Injection attacks, RPTD method outperformed the other three proposed methods. In a future work, we shall further investigate why a method performs best for some attacks datasets and not for others, potentially through machine-learning interpretability studies.

Comparing Time Costs

Another aspect of comparing the performance of the proposed methods is to look at the time required to train, retrain, or execute (for inference) the ensembles. In the CITD and RITD methods, since the de-activations are carried out during training, the number of AEs being trained are lesser than in CPTD and RPTD methods. This leads to savings in terms of training/re-training time costs as well as in terms of computational costs. For example, CPTD trains all AEs in the ensemble (except the final output layer AE) in about 1.4ms per training sample. CITD on the other hand consumes an average of 0.8ms per training sample (almost half the time taken by CPTD).

In assessing inference time costs of the proposed methods, note that ensembles of equal size, say 7, have the same number of AEs in the ensemble and the dimensions of the final output layer AE are the also the same for CPTD, CITD, RPTD and RITD methods. Hence, the inference (testing) time cost would be similar. However, the advantages of the proposed methods over the case of no de-activations is evident when one looks at their testing time costs. While the baseline ensemble (with no de-activations) takes 1.1ms per sample, the proposed methods (with final ensembles of size 7 after the de-activations) consume 0.26ms per sample, i.e., a speedup of 4.2×. Hence, the proposed methods of learning smaller ensembles via de-activations provide training, re-training and inference time cost savings.

4.4.4 Comparison with Other Methods (on NBaIoT Datasets)

In addition to Kitsune datasets, we evaluate the performance of the proposed methods using the NBaIoT datasets of [175]. An ensemble is learnt for each device using its benign samples. The parameters utilized for the respective methods are as follows: $\beta = -0.5$ (in criteria-based methods), $N\_DeAc\_rounds = 10$ (for in-training de-activations), $DeAc\_Ratio = 0.2$ per round for RITD and $DeAc\_Ratio = 0.7$ for RPTD. Tables 4.9 and 4.10 present the AUC and EER scores (respectively) of the proposed methods and of the baseline method that has ensembles with no de-activations.

As one could observe in Table 4.9, the proposed CITD method performed as good as

the baseline ensemble in which there are no de-activated AEs. This is despite the reduced ensemble size and lower number of AEs in CITD.

Looking at the EER scores in Table 4.10, we find that the proposed methods achieved lowest EER scores for some devices whereas for other devices the ensembles with no de-activations yielded lower EERs. The mean EER of ensembles with no de-activations (*No DeAcs*) was 0.0009 whereas CITD achieved a mean EER of 0.0165. Although the proposed methods yielded higher EERs on some devices, the performance in terms of AUC scores was still at par or better than the baseline ensembles with no de-activations.

We compare the performance of the proposed methods against the results of two recently published works on NBaIoT datasets: [187] and [175]. The works of [175, 187] are also unsupervised learning-based approaches where the malicious traffic samples are detected as anomalies from the learnt benign traffic patterns. Considering comparisons in terms of mean False Positive Rates (mFPR) (see Table 4.11), the deep autoencoders-based intrusion detection of [175] achieved mFPR of 0.007 while the CPTD and CITD methods achieved mFPR of 0.344 and 0.016 respectively. In addition, [175] tested other methods such as Isolation Forest (Iso. For.) and Local Outlier Factor (LOF) which also yielded higher mFPRs of 0.027 and 0.086 respectively.

Another work based on the NBaIoT dataset is of [187] who investigated SVMs and Isolation Forests trained over a subset of the statistical features provided by the datasets. The features were selected based on metrics such as entropy, variance and Hopkins statistic. As shown in Table 4.12, the proposed method CITD outperforms the best models of [187] for all devices.

4.5 Concluding Remarks

In this chapter, we proposed, evaluated and compared four methods to build efficient and adaptive ensembles of neural networks-based AEs for intrusion detection in internet of things such as surveillance cameras. In doing so, the proposed methods orchestrate and conduct de-activations of constituent AEs in the ensemble. These methods differ in their approach to select AEs for de-activation (i.e., criteria-based or random) and also differ in

terms of when the de-activations are performed (i.e., post-training or in-training). In the criteria-based de-activation approach, the training performance of an ensemble's AEs are utilised to adaptively de-activate some of them *in-training* or *post-training*. In the random de-activation approach, the ensemble's AEs are randomly de-activated post- or in-training.

Through comprehensive experiments on realistic IoT datasets, we showed that the proposed methods enable achieving a less costlier ensemble of AEs (in terms of computation, training, re-training and inference time costs) at satisfactory levels of detection performance (in terms of AUC and EER scores). In massive IoT environments where an IDS is composed of ensembles of AEs for each device or device-type, our methods provide for more efficient ensembles which could be scaled up or down adaptively, paving the path towards a scalable and efficient intrusion detection system or service that could be deployed on-device or on-edge.

Table 4.6: Performance Comparison of Medium-sized Ensembles learnt through RPTD and RITD methods)

Attack			**ARP MitM**				**Fuzzing**	
Method / Metric	AUC	TPR	EER	#AEs	AUC	TPR	EER	#AEs
No De-Ac	0.86	0.06	0.187	23	0.296	0.024	0.636	23
RPTD (0.5)	0.687	0.107	0.387	12	0.356	0.024	0.599	12
RITD (0.1)	0.837	0.013	0.198	12	0.499	0.024	0.503	12
Attack			Mirai				OS Scan	
Method / Metric	AUC	TPR	EER	#AEs	AUC	TPR	EER	#AEs
No De-Ac	0.943	0.878	0.105	23	1	1	0	23
RPTD (0.5)	0.944	0.878	0.104	12	0.999	1	0	12
RITD (0.1)	0.940	0.878	0.109	12	1	1	0	12
Attack			SSDP Flooding				SSL Renegotiation	
Method / Metric	AUC	TPR	EER	#AEs	AUC	TPR	EER	#AEs
No De-Ac	0.999	0.999	0.001	23	0.98	0.167	0.067	23
RPTD (0.5)	0.999	0.999	0.001	12	0.972	0.092	0.094	12
RITD (0.1)	0.999	0.999	0.001	12	0.979	0.091	0.054	12
Attack			SYN DoS				Video Inj.	
Method / Metric	AUC	TPR	EER	#AEs	AUC	TPR	EER	#AEs
No De-Ac	0.976	0	0.062	23	0.890	0.009	0.179	23
RPTD (0.5)	0.522	0	0.464	12	0.888	0.073	0.166	12
RITD (0.1)	0.861	0	0.158	12	0.718	7.81E-05	0.358	12
Attack			Wiretap					
Method / Metric	AUC	TPR	EER	#AEs				
No De-Ac	0.989	0.801	0.045	23				
RPTD (0.5)	0.985	0.041	0.048	12				
RITD (0.1)	0.827	0.367	0.314	12				

Table 4.7: Performance Comparison of Small-sized Ensembles learnt through RPTD and RITD methods

Attack	**ARP MitM**				**Fuzzing**			
Method / Metric	AUC	TPR	EER	#AEs	AUC	TPR	EER	#AEs
No De-Ac	0.86	0.06	0.187	23	0.296	0.024	0.636	23
RPTD (0.5)	0.589	0.032	0.435	3	0.419	0.019	0.555	3
RITD (0.1)	0.601	0.031	0.432	3	0.529	0.024	0.511	3
Attack	**Mirai**				**OS Scan**			
Method / Metric	AUC	TPR	EER	#AEs	AUC	TPR	EER	#AEs
No De-Ac	0.943	0.878	0.105	23	1	1	0	23
RPTD (0.5)	0.942	0.878	0.107	3	0.999	1	9.00E-05	3
RITD (0.1)	0.252	0.001	0.791	3	0.999	0.994	0.005	3
Attack	**SSDP Flooding**				**SSL Renegotiation**			
Method / Metric	AUC	TPR	EER	#AEs	AUC	TPR	EER	#AEs
No De-Ac	0.999	0.999	0.001	23	0.98	0.167	0.067	23
RPTD (0.5)	0.999	0.997	0.003	3	0.901	0.092	0.150	3
RITD (0.1)	0.999	0.919	0.002	3	0.510	0.010	0.544	3
Attack	**SYN DoS**				**Video Inj.**			
Method / Metric	AUC	TPR	EER	#AEs	AUC	TPR	EER	#AEs
No De-Ac	0.976	0	0.062	23	0.89	0.009	0.179	23
RPTD (0.5)	0.363	0	0.603	3	0.780	0.320	0.300	3
RITD (0.1)	0.398	0	0.570	3	0.556	0.041	0.468	3
Attack	**Wiretap**							
Method / Metric	AUC	TPR	EER	#AEs				
No De-Ac	0.989	0.801	0.045	23				
RPTD (0.5)	0.741	0.061	0.329	3				
RITD (0.1)	0.716	0.390	0.363	3				

DeAc_Ratio in the brackets, for RITD, applies per round; TPR is at FPR= 0.001

Table 4.8: Performance Comparison (AUC Scores) of CDA, RDA and Other Methods on Kitsune Datasets

Method \Attacks	**ARP MitM**	**Fuzzing**	**Mirai**
No DeAcs	0.860	0.296	0.943
CPTD	*0.880*	0.342	0.945
CITD	**0.890**	0.336	*0.947*
RPTD	0.604	0.398	0.944
RITD	0.601	0.523	0.944
Kitsune (m=10) [180]	0.584	**0.999**	**0.999**
Isolation Forest (in [180])	0.598	*0.958*	0.526
GMM (in [180])	0.767	**0.999**	**0.999**
Method \Attacks	**OSScan**	**SSDPFlood**	**SSLRen**
No DeAcs	**1.0**	0.999	*0.980*
CPTD	*0.999*	0.999	0.967
CITD	*0.999*	0.999	0.956
RPTD	*0.999*	0.999	0.901
RITD	**1.0**	0.999	**0.987**
Kitsune (m=10) [180]	0.948	0.999	0.974
Isolation Forest (in [180])	0.907	0.948	0.872
GMM (in [180])	0.949	0.999	0.965
Method \Attacks	**SYNDoS**	**Video Inj.**	**Wiretap**
No DeAcs	**0.976**	**0.890**	**0.989**
CPTD	0.896	0.878	*0.985*
CITD	0.520	0.611	0.842
RPTD	0.552	*0.884*	0.948
RITD	0.729	0.681	0.926
Kitsune (m=10) [180]	*0.950*	0.854	0.882
Isolation Forest (in [180])	0.587	0.517	0.574
GMM (in [180])	0.834	0.873	0.879

The values in bold and italics are the highest and second highest scores respectively

Table 4.9: Performance Evaluation (AUC Scores) of CDA, RDA and No De-activations on NBaIoT Datasets

Dataset	No DeAcs	CPTD	CITD	RPTD	RITD
Dev 1	*0.998*	**0.999**	**0.999**	**0.999**	0.984
Dev 2	**0.999**	0.566	**0.999**	*0.612*	**0.999**
Dev 3	**0.999**	0.367	*0.981*	0.398	**0.999**
Dev 4	**0.999**	*0.898*	**0.999**	0.614	**0.999**
Dev 5	**0.999**	0.766	**0.999**	0.612	*0.871*
Dev 6	**0.999**	0.701	**0.999**	0.684	*0.993*
Dev 7	**0.999**	0.505	**0.999**	*0.796*	**0.999**
Dev 8	**0.999**	*0.951*	**0.999**	0.746	0.915
Dev 9	**0.999**	*0.977*	**0.999**	*0.977*	**0.999**
MeanAUC	**0.999**	0.748	*0.997*	0.715	0.973

The values in bold and italics are the highest and second highest scores respectively

Table 4.10: Performance Evaluation (EER Scores) of CDA, RDA and No De-activations on NBaIoT Datasets

Dataset	No DeAcs	CPTD	CITD	RPTD	RITD
Dev1	*0.0027*	**0.0019**	0.0034	0.0019	0.0202
Dev2	*0.0003*	0.4903	**0**	0.4923	**0**
Dev3	*0.0012*	0.6024	0.1134	0.633	**0.0009**
Dev4	**0.0004**	*0.3335*	**0.0004**	0.4429	**0.0004**
Dev5	**0.0004**	0.4867	*0.0029*	0.5397	0.1186
Dev6	**0.0004**	0.5229	*0.0031*	0.5309	0.0233
Dev7	*0.0011*	0.6719	0.0018	0.1567	**0.0006**
Dev8	**0.0012**	0.0513	*0.0029*	0.4891	0.0905
Dev9	**0.0005**	0.0236	0.0204	0.0236	*0.0195*
MeanEER	**0.0009**	0.3538	*0.0165*	0.3678	0.0304

The values in bold and italics are the highest and second highest scores respectively

Table 4.11: Mean False Positive Rates of CPTD, CITD with those reported by [175] on NBaIoT Datasets

Metric	**CPTD**	**CITD**	**NBaIoT [175]**	**Isolation Forest**	**LOF**
Mean FPR (%)	0.344	0.016	0.007	0.027	0.086

Table 4.12: Precision scores (malicious class) Comparison of CITD and results reported by [187] on NBaIoT Datasets

Dataset	**Dev 1**	**Dev 2**	**Dev 3**	**Dev 4**	**Dev 5**
No DeAcs	**0.999**	**1.0**	**0.999**	**1.0**	**1.0**
CITD	**0.999**	**1.0**	*0.993*	**1.0**	*0.999*
[187]	*0.995*	*0.998*	N/A	*0.947*	0.991
Dataset	**Dev 6**	**Dev 7**	**Dev 8**	**Dev 9**	
No DeAcs	**1.0**	**0.999**	**1.0**	**1.0**	
CITD	*0.999*	**0.999**	*0.999*	*0.999*	
[187]	0.986	N/A	0.994	0.997	

The values in bold and italics are the highest and second highest scores respectively

Chapter 5

A Lightweight Defense Method against Adversarial Patches-based Attacks on Automated Vehicle Make and Model Recognition Systems

In smart cities, connected and automated surveillance systems play an essential role in ensuring safety and security of life, property, critical infrastructures and cyber-physical systems. The recent trend of such surveillance systems has been to embrace the use of advanced deep learning models such as convolutional neural networks for the task of detection, monitoring or tracking. In this chapter, we focus on the security of an automated surveillance system that is responsible for vehicle make and model recognition (VMMR). We introduce an adversarial attack against such VMMR systems through adversarially learnt patches. We demonstrate the effectiveness of the developed adversarial patches against VMMR through experimental evaluations on a real-world vehicle surveillance dataset. The developed adversarial patches achieve reductions of up to 48% in VMMR recall scores. In addition, we propose a lightweight defense method called *SIHFR* (stands for *Symmetric Image-Half Flip and Replace*) to eliminate the effect of adversarial patches on VMMR performance. Through experimental evaluations, we investigate the robustness of the proposed defense method under varying patch placement strategies and patch sizes. The proposed defense method adds a minimal overhead of less than 2ms per image (on average) and succeeds in enhancing VMMR performance by up to 69.28%. It is hoped that

this work shall guide future studies in developing smart city VMMR surveillance systems that are robust to adversarial learning-based cyber-physical attacks.

5.1 Introduction

The surveillance systems such as automated Vehicle Make and Model Recognition (VMMR) systems are essential components in smart cities and intelligent transportation systems to aid in ensuring safety and security of life, property, critical infrastructures and in security management of cyber-physical systems. The state-of-the-art VMMR systems are based on advanced deep learning models such as convolutional neural networks (CNNs). These are highly vulnerable to a new kind of cyber-physical attack that leverages adversarial machine learning. Recent studies have shown that the CNNs could be tricked or fooled to evade detection or cause mis-classification [1,116,141,254]. One of the ways in which this is achieved involves crafting or modifying the inputs through adversarially learnt patterns printed or patched on the objects, presenting a unique kind of cyber-physical attack. For example, the work of [240] developed adversarial posters and stickers to cause object detectors to not detect stop signs which is a potentially lethal attack against connected and autonomous vehicles. As another example, CNNs trained to detect and recognize persons were not able to detect them due to the placement of adversarial patches on those persons [254].

To the best of our knowledge, no prior studies have investigated the adversarial robustness of CNNs-based VMMR systems against such attacks. In this chapter, we study adversarial patches-based attacks specifically targeted against VMMR systems (see Figure 5.1), a problem that has not been addressed in the literature to the best of our knowledge. Moreover, we propose a lightweight defense to mitigate the effect of adversarial patches, leveraging symmetry in vehicles' frontal (or rear) faces. The proposed defense method does not require additional hardware. It can be deployed practically as a complementary component working in tandem with, or incorporated into, an automated VMMR software. Possible adopters of this technology may include smart cities (for automated surveillance), security agencies and traffic analysts.

In areas where security is an important concern, automated VMMR systems find their applications in recognizing the makes and models of vehicles from the images captured

by surveillance cameras [68, 92]. Examples of such areas include parking lots of airports, malls, checkpoints, critical facilities, etc. Moreover, automated VMMR systems serve as efficient and fast solutions to hunt suspects or targets, e.g., in aiding police to search for a vehicle of certain make and model. Malicious entities could leverage adversarial patches to attack and circumvent VMMR systems to gain unauthorized access or avoid being found. There could be two broad types of possible attacks: (i) impersonation and (ii) dodging. In the former, an adversary's goal is to impersonate a particular vehicle make and model by tricking the VMMR system. In the latter, an adversary's goal is to evade correct recognition of its make and/or model by tricking the VMMR system to recognize it as any other make and/or model.

The specific research questions that motivate this study include: (i) How feasible is it to launch adversarial attacks against CNN-based VMMR systems, (ii) How successful could adversarial attacks be in tricking a VMMR system to mis-classify vehicles (in terms of impact on precision and recall scores), and (iii) Could we develop a lightweight defense method, without requiring to modify or re-train the CNN model, to eliminate or reduce the influence of adversarial patches? Based on these, the main contributions this chapter makes are as follows: (i) study the learning of adversarial patches to fool CNN-based VMMR systems, (ii) investigate the impact of adversarial attacks based on these learnt patches in reducing precision and recall scores of a CNN-based VMMR system, (iii) design, develop and evaluate a lightweight defense method to improve VMMR system's performance under adversarial patches-based attacks, (iv) investigate the robustness of the proposed defense method under varying patch placement strategies, and (v) demonstrate how the proposed defense method outperforms a state-of-the-art defense method.

5.2 Adversarial Patches-based Attacks against VMMR Systems

In this section, we introduce adversarial patches that could be printed and placed on or around vehicles to fool the VMMR surveillance systems. We describe in detail the process of learning these adversarial patches. Unlike the prior works such as [230,240,254], our work

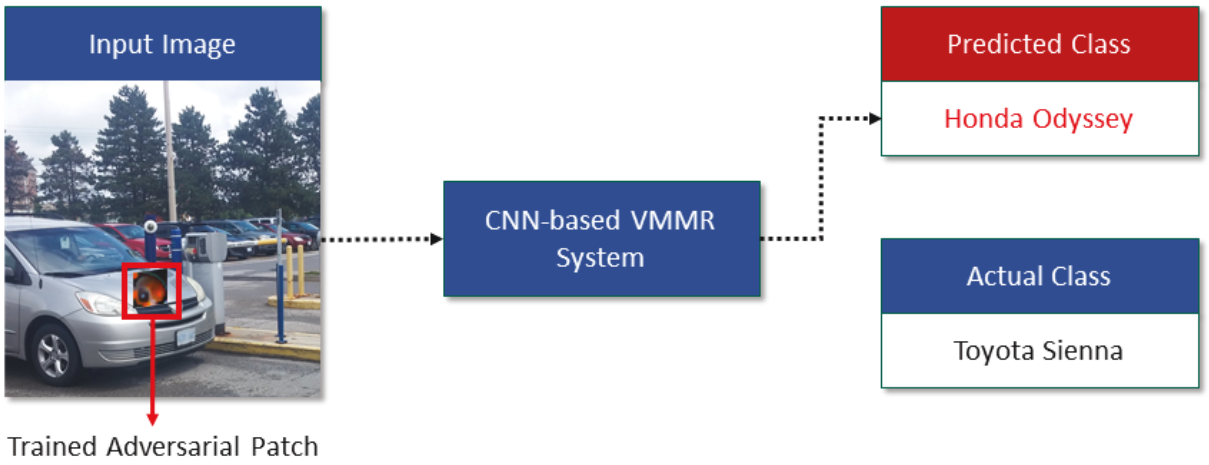


Figure 5.1: Illustration of a cyber-physical attack in action: using an adversarial patch placed on a vehicle to fool a VMMR surveillance system that is based on deep learning models such as CNNs. In this paper, we show how such adversarial patches could trick the VMMR system to mis-classify vehicles to wrong classes in order to evade detection or impersonate as another vehicle. In addition, we propose a lightweight defense method to mitigate such attacks.

is the first, to the best of our knowledge, that targets VMMR systems. Unlike stop signs or persons, vehicles not only differ in appearance (multiplicity) but have many similarities as well (inter-class and intra-class ambiguity). In brief, we learn adversarial patches which when placed on or around a vehicle lowers the VMMR classification accuracy.

The adversarial patches-based attacks against VMMR systems could be launched by physically placing the printed patches on the vehicles or by digitally placing the patches on the captured images. The digital placement of patches could be achieved by compromising the surveillance camera network, e.g., through video injection attacks or man-in-the-middle type of attacks. Many works have studied the problem of network intrusions in IoT and camera networks (e.g., [**?**, 29, 70, 78, 103, 146, 155, 158, 163, 224, 233, 252, 253]). The focus of this chapter is on defending against attacks launched by placing the adversarial patches on the vehicles regardless of whether it is done physically or digitally. Nonetheless, in training the adversarial patches, the patch transformation and update module (as described in Sections 5.2.2 and 5.2.1 respectively) factors into account real-world considerations of physical patch placement and appearance.

In what follows, we describe the method to train and learn such adversarial patches. The overview of the process to learn the adversarial patches in shown in Figure 5.2. The main modules are: *Patch Generation & Update*, *Patch Transformation & Placement*, *Model Execution*, *Loss Calculation & Backpropagation*, as described further below. Table 5.1 describes the main notations used in the chapter.

5.2.1 Patch Generation and Update

The adversarial patch learning process starts with a generated patch that gets updated through the learning process. The initial patch may either be generated with random values or an adversarial patch trained against another object detector (e.g., person detector) could be used. The patch produced by the former approach is termed as a *random initial patch*, whereas the patch used in the latter approach is referred to as a *pre-trained initial patch*. Regardless of the approach to generate the initial patch, the patch undergoes updates through the training process, updating its values based on the back-propagated gradients (that are described in Section 5.2.4).

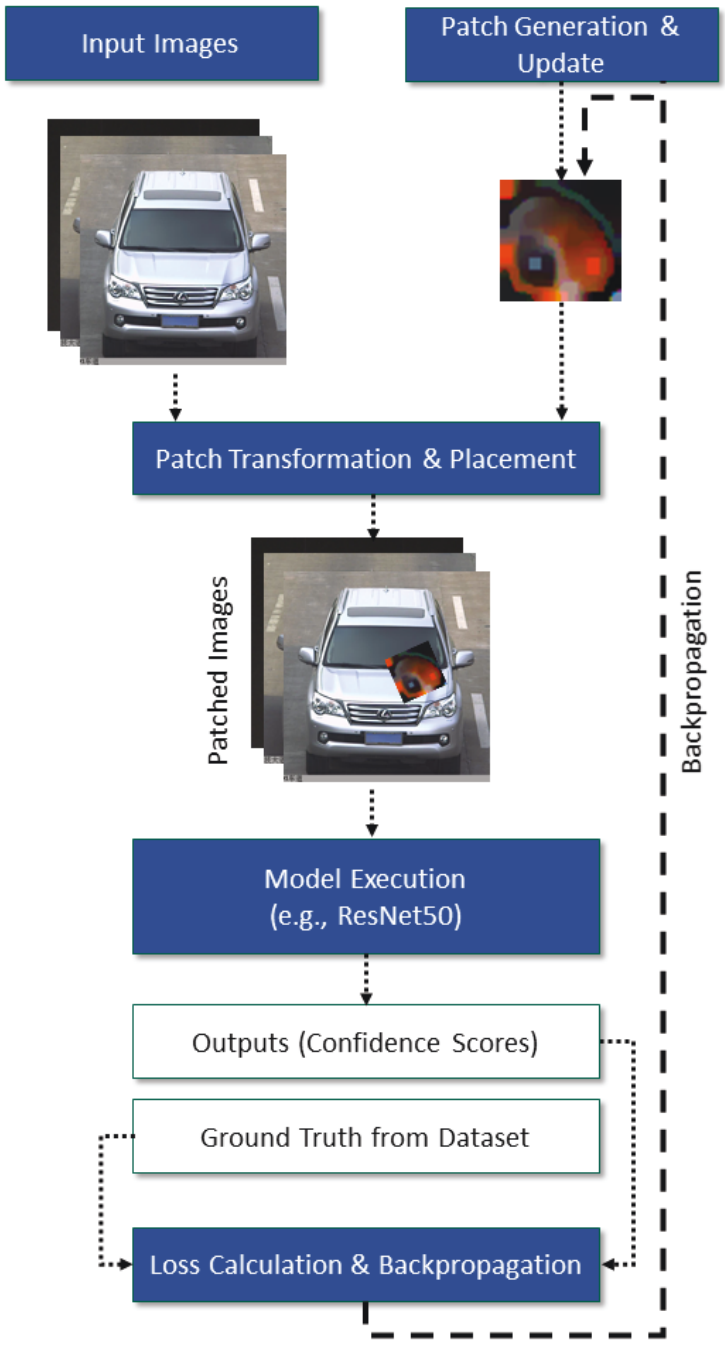


Figure 5.2: Overview of the adversarial patch learning process targeting a CNN-based VMMR System

Table 5.1: Table of Notations

Symbol	Description
$\mathcal{L}_{CC}$	The highest confidence score in a list of outputs from the model execution module
$\mathcal{L}_{PNP}$	Non-printability loss of a learnt patch
$\mathcal{L}_{PS}$	Patch smoothness loss
$\mathcal{L}$	The overall loss function (Equation 5.3)
$I_{i,j}$	Refers to a pixel at the coordinate (i, j) of image I
I_L and I_R	The left and right halves of an image I, respectively
$TV(I)$	The total variation [170], given an image I, as expressed in Equation 5.4
'-TL'	Patch placed at the top-left of images in training
'-RL'	Patch placed at random locations on images in training
'(TL)'	Patch placed at the top-left of images in testing
'(RL)'	Patch placed at random locations on images in testing

5.2.2 Patch Transformation and Placement

The initial and updated patches are transformed and placed on input images in this module. The transformations include scaling, rotation, brightness/contrast adjustments, and noise addition. These transformations have to be such that it is possible to perform gradient computations during backpropagation to update the patch values [254]. The transformed patches are then placed on the input images at specified or random locations in the image. We study and evaluate, in Section 5.5.1, both patch placement approaches (specified or random) and their effectiveness in reducing the VMMR performance.

The patch transformations help in learning adversarial patches that are robust to inevitable factors experienced in real-world applications. An example real-world application is a camera-based VMMR system that captures the images of incoming or exiting vehicles and the adversarial patches may be placed on or around these vehicles.

To elaborate, when a printed adversarial patch is placed on the vehicle, its appearance to the camera may change due to changing conditions such as lighting. Since the vehicles may be at different viewing angles to the camera, the viewing angles of the patches may vary as well. In addition, since vehicles in captured images vary in size, the relative size of patches and vehicles varies. Moreover, there may be noise or blur introduced by the camera capturing the input images [254]. Also, an adversary may place the patches at different locations on or around the vehicle. Hence, the patch transformation and placement module incorporates the influences of such factors in the patch training process, thereby achieving robustness to these factors.

5.2.3 Model Execution

In this module, a deep learning model (CNN) such as ResNet50 [135] that has been trained to achieve high classifications rates for VMMR is used. The patched images from Patch Transformations and Placement module are fed into the Module Execution module (as depicted in Figure 5.2). It is worth mentioning that any deep learning model that produces a list of confidence scores could be utilized in place of ResNet50. We choose ResNet50 for VMMR due to its demonstrated success in achieving high accuracy levels for image classification [135].

Each patched image makes a forward pass through the model, producing a list of confidence scores for each of the classes in the considered dataset. The class winning the highest confidence score that is above a certain acceptable threshold is regarded as the predicted class of the vehicle in the input image. It is worth noting that an attacker doesn't need to know the internal details or architecture of the CNN or other network performing the VMMR. The adversarial patches are trained based on the confidence scores output by the VMMR model.

5.2.4 Loss Calculation and Backpropagation

The loss calculation module determines the values for the loss functions described below, using the ground truth data of input images and the outputs from the model execution module. In this work, we adopt the following loss functions, inspired by the approach of [254]: (i) Class Confidence Loss ($\mathcal{L}_{CC}$), (ii) Non-Printability Loss ($\mathcal{L}_{PNP}$), and (iii) Patch Smoothness Loss ($\mathcal{L}_{PS}$). The three loss functions are described below.

- $\mathcal{L}_{CC}$: The highest confidence score in a list of outputs from the model execution module. The goal is to minimize the confidence score corresponding to the ground truth class to achieve the objective of the adversarial patch, i.e., to miss-classify the given vehicle's image as any other vehicle class.

- $\mathcal{L}_{PNP}$: The non-printability loss measures how far the adversarial patch's pixel color values are from the given set of commonly printable colors [230]. It is formulated as:

$$\mathcal{L}_{PNP} = \sum_{p_{i,j} \in P} \left(\min_{c \in C} |p_{i,j} - c| \right. \tag{5.1}$$

  Here, c refers to a color from the set of commonly printable colors given by C whereas $p_{i,j}$ refers to a pixel at (i, j)-th coordinate in the adversarial patch P.

- $\mathcal{L}_{PS}$: The patch smoothness loss measures the total variation [170] of the patch. Lower the total variation, higher would be the patch smoothness and vice-versa. The goal in this work is to learn adversarial patches that are smoother, following the approach of [230,254], so that the patched images resemble closely the natural images

in terms of smoothly gradually changing colors. Additionally, as noted in [230], cameras may not be able to properly capture high variations in a patch's adjacent pixels due to sampling noise. Hence, patches with low smoothness loss, i.e. high smoothness, are more effective for real-world applications. The $\mathcal{L}_{PS}$ is formulated as:

$$\mathcal{L}_{PS} = \sum_{p_{i,j} \in P} \Big((p_{i,j} - p_{i+1,j})^2 + (p_{i,j} - p_{i,j+1})^2 \tag{5.2}$$

The overall loss function is a weighted aggregate of the above three loss functions, as given below, where the values of α, β and γ are chosen through empirical evaluations:

$$\mathcal{L} = \alpha \mathcal{L}_{CC} + \beta \mathcal{L}_{PNP} + \gamma \mathcal{L}_{PS} \tag{5.3}$$

The loss function is minimized through backpropagation and Adam optimizer [150]. In doing so, only the patch values are updated while the weights of the deep learning model (e.g., ResNet50) are frozen.

5.3 Proposed Defense Method: Symmetric Image-Half Flip and Replace (SIHFR)

In this section, we introduce the proposed *Symmetric Image-Half Flip and Replace (SIHFR)* method designed and developed to defend against adversarial patches-based attacks on VMMR surveillance systems. The proposed method is based on certain observations on the nature of images captured by surveillance cameras in VMMR systems. Specifically, in this work we consider VMMR systems that capture images of incoming or outgoing vehicles such that the captured images have one vehicle each. We assume that the images which are going to be fed to VMMR algorithms have the vehicles at their centers, i.e., the vehicles are centrally aligned with respect to the input images. This is a reasonable assumption since such images are commonly found in VMMR deployment scenarios such as checkpoints, entrances or exits of secured areas [92, 235]. In fact, the dataset used in this work was collected from a real-world surveillance camera capturing incoming vehicles, and contains images of the aforementioned nature. In the current work, we focus on frontal view images only, however our method is applicable to rear view images as well.

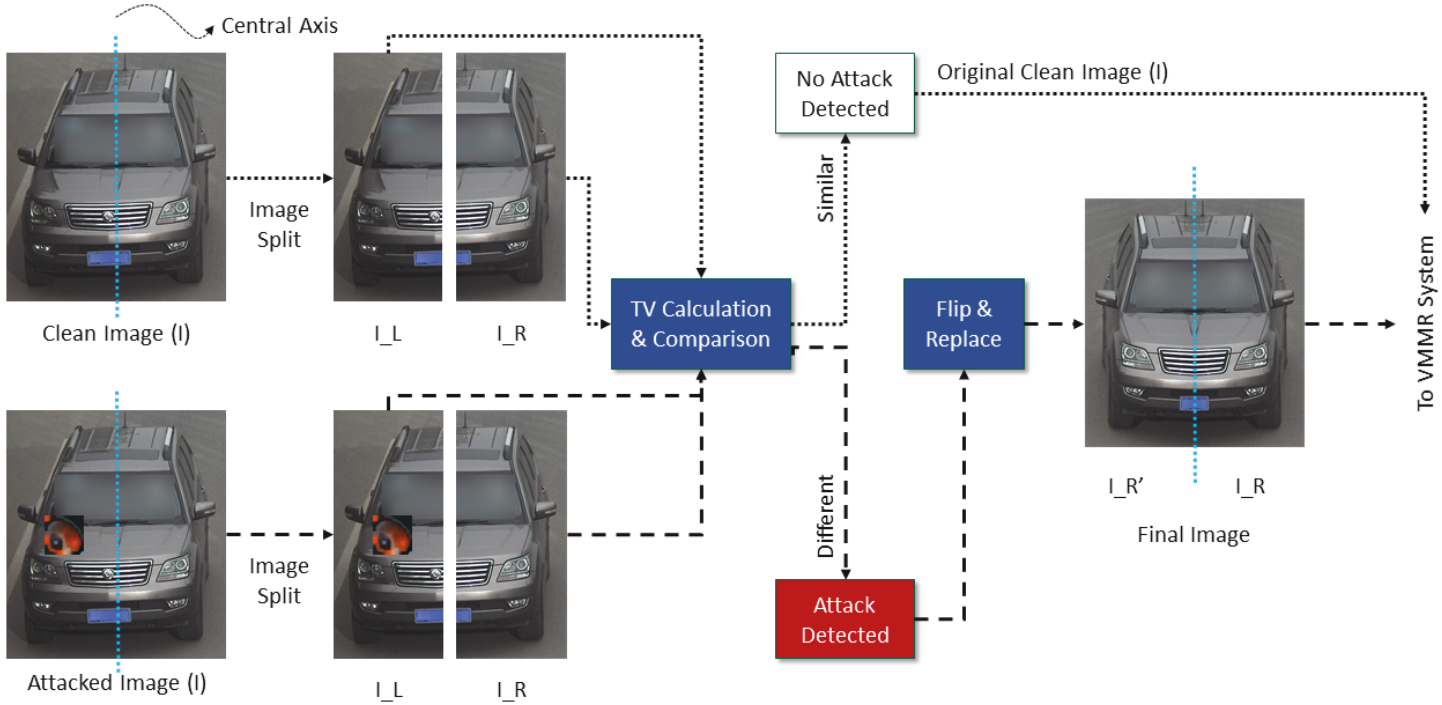


Figure 5.3: A flowchart of the proposed SIHFR method to defend against adversarial patches-based attacks on VMMR systems. In the final image, $I_{R'}$ refers to the horizontally flipped I_R which was selected to replace I_L because $TV(I_L) > TV(I_R)$.

The SIHFR method has been designed to meet the following requirements. First, it shall not require re-training or fine-tuning of the CNN-based VMMR models. This is essential because re-training and fine-tuning of CNN-based VMMR models is an expensive process in terms of computational and time costs. Second, it shall eliminate or reduce the effect of adversarial patches on the VMMR scores. Third, it shall be lightweight and shall not add significant overheads in terms of computational or time resources, both in training and execution phases.

The working principle of the proposed SIHFR method is based on the following observations. The vehicle images captured in the aforementioned conditions are found to be symmetric around a vertical central axis, hereby referred to as the central axis, even under slight variations in viewpoints of the incoming vehicles, as illustrated in Figure 5.3. An adversary could place the adversarial patch any where on the vehicle such that the patch appears on the left or right of the central axis of symmetry. It may also be possible that the patch is placed along the central axis such that it partly appears on both sides of the axis.

Since the left and right image halves (split by the central axis) are symmetric to each other, they exhibit similar values of Total Variation (TV) [170], under normal conditions, i.e., in absence of any adversarial patches. Given an image (or image part) I, the value of total variation, $TV(I)$, is calculated as per Equation 5.4, where $I_{i,j}$ refers to a pixel at the coordinate (i, j) of I. When an adversarial patch appears on either image half, the TV value of that half will be different from that of the other image half. We utilize this difference in TV to identify the image half that is 'clean' and the one that is potentially 'modified' with the adversarial patch. The clean half is selected, horizontally flipped and then copied onto the other half. In this manner, the method gets rid of the adversarial patch which was placed on the 'modified' image half. Figure 5.3 presents a flowchart of the propose method with a sample clean image (with no adversarial patch) and an attacked image (with an adversarial patch).

The *'TV Calculation and Comparison'* step (Figure 5.3) calculates and compares the TV values of the left and right image halves I_L and I_R, respectively. If the TV values are similar, no adversarial patches are detected and so the original input image is passed on to the VMMR system. On the other hand, if the TV values are different, an adversarial

patch attack is detected. In the *'Flip and Replace'* step, the image half with a lower TV value is selected as the clean half which is flipped to replace the attacked half. This is based on the observation, using training images, that the image half with an adversarial patch has a higher TV value, despite the fact that the adversarial patch training takes into account the goal of maximizing patch smoothness (as described in Section 5.2.4) to reduce the total variation.

$$TV(I) = \sum_{I_{i,j} \in I} \Big((I_{i,j} - I_{i+1,j})^2 + (I_{i,j} - I_{i,j+1})^2 \tag{5.4}$$

5.4 Experimental Setup

In this section, we present the experimental setup used in training and testing evaluations of adversarial patches targeted against VMMR systems and of the proposed SIHFR defense method. Beginning with a description of the real-world dataset used in this work and describing the performance metrics that shall be used in the evaluations, we then present the training process of three groups of adversarial patches developed and evaluated in this work.

5.4.1 Dataset

In order to evaluate the effectiveness of adversarial patches against VMMR and of the SIHFR defense method, we choose the CompCars Surveillance Dataset [265]. Though there are other publicly available datasets for VMMR [92], we found [265]'s dataset to be most representative of the real world scenarios this work aims to target (see Table 2.2). Its composition is such that it allows one to train a VMMR model that deals with the challenges mentioned in Section 2.3.3 (e.g., multiplicity and inter- or intra-class ambiguities). The adversarial patches developed in this work are targeted against such a robust VMMR system.

The images in CompCars Surveillance Dataset were collected by on-road surveillance cameras that capture the oncoming vehicles' frontal views. The images were taken under

varying lighting conditions. The dataset has 281 classes with a total of 31,149 images for training and 13,334 images for testing. It also contains make-model classes that have different versions and appearances over different years.

In real-world applications, a VMMR system could encounter different types of vehicles. As such, the selected dataset comprises of the following different vehicle types: sedan, hatchback, fastback, SUV, MPV, minibus, estate, crossover, convertible, hardtop convertible, sports, and pickup [265].

5.4.2 Performance Metrics

The performance of a VMMR system, in terms of how good or bad it is in classifying different make-model classes, is assessed based on the following metrics: precision, recall, and F1 scores, averaged across all classes in the dataset. The effectiveness of an adversarial patch in fooling the VMMR system could then be assessed by the amount of reduction it causes in these VMMR scores. The higher the reduction, more effective is the adversarial patch. Amongst these scores, reduction in recall scores is the most important for a non-targeted attack [254].

Below, we provide classwise definitions for these metrics, where TP_i, FP_i and FN_i refer to the True Positives, False Positives and False Negatives with respect to a class-i, respectively. The classwise metrics are averaged to obtain average precision, recall and F1 scores that represent the overall VMMR performance.

- Attack Detection Rate: the ratio of correctly detected adversarial patches-based attacks to the total number of attacked test images.
- Precision: For a class i, its Precision score P_i is:

$$P_i = \frac{TP_i}{TP_i + FP_i} \tag{5.5}$$

- Recall: For a class i, its Recall score R_i is:

$$R_i = \frac{TP_i}{TP_i + FN_i} \tag{5.6}$$

- F1-Score: a harmonic average of Precision and Recall scores

$$\textit{F1-Score}_i = \frac{2 \cdot P_i \cdot R_i}{P_i + R_i} \tag{5.7}$$

Then, a reduction in these metrics (especially the Recall metric) caused by placement of the proposed adversarial patches on the images is measured in terms of a percentage decrease that reflects the attack's success rate. The success of a defense method on the other hand is assessed based on the attack detection rate and on the improvements in scores of the attacked VMMR system.

5.4.3 Training of Adversarial Patches

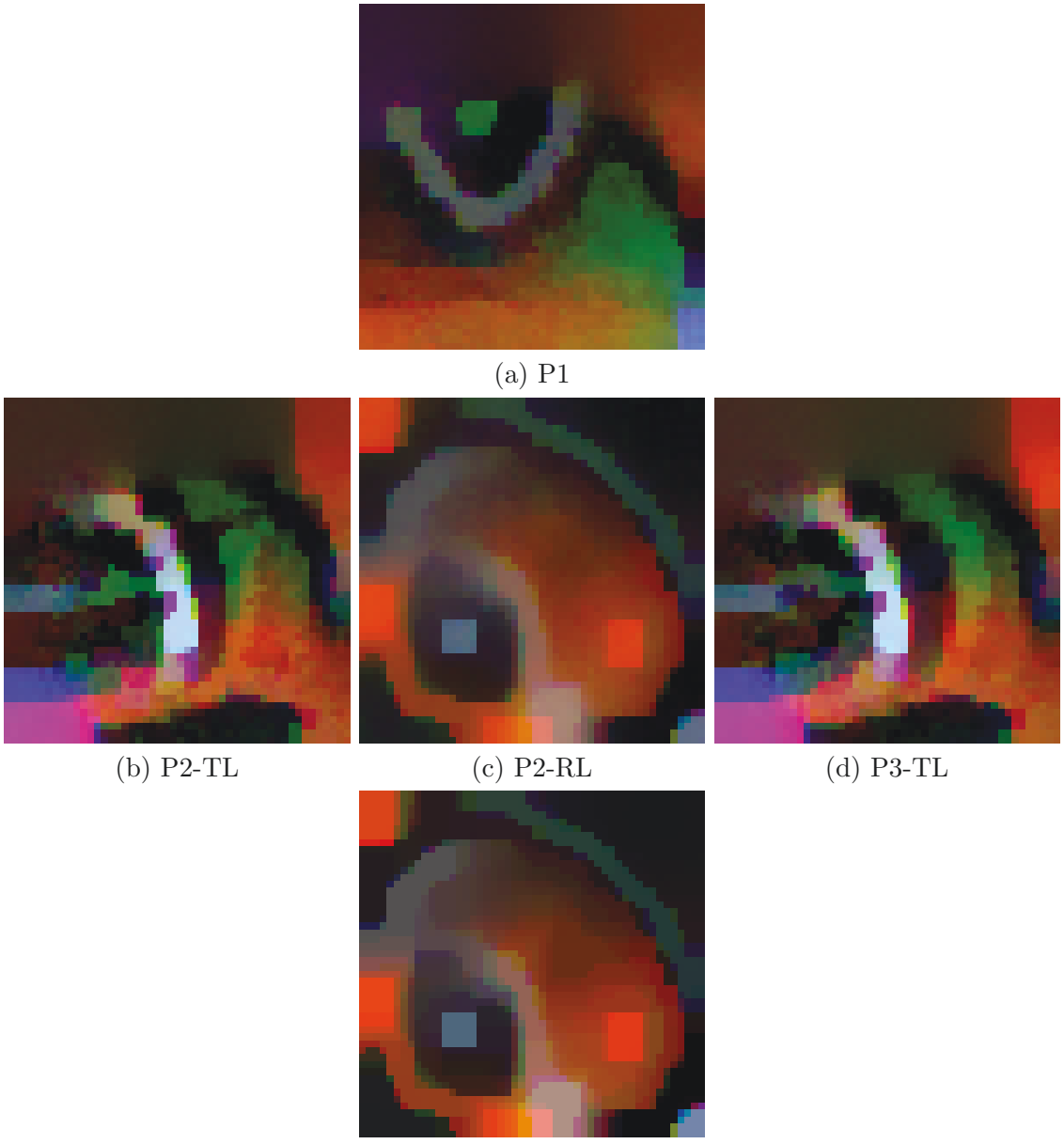
The adversarial patches are learnt following the procedure described in Section 5.2. In this work, we study three groups of adversarial patches which differ in the number of epochs they're trained for, or in the initial patch configuration, or in the weights used in the loss function of Equation 5.3.

Group 1

In the first group, we train three patches, setting the weights $\alpha = \beta = \gamma = 1$ for the loss function of Equation 5.3 and starting from a blank gray initial patch. The three patches P1, P2 and P3 differ in the number of epochs they're trained for:

- $P1$: trained for a lower number of epochs (45) and smaller batch size (25).
- $P2$: trained for a higher number of epochs (200) and a larger batch size (100).
- $P3$: trained for a higher number of epochs (300) and a large batch size (100).

The patches P2 and P3 are trained under two patch placement settings: (i) *TL*: the placement of patches is at a fixed location on input images, i.e., the top-left, and (ii) *RL*: the patches are placed at random locations on the input images. In testing, we evaluate



(a) P1

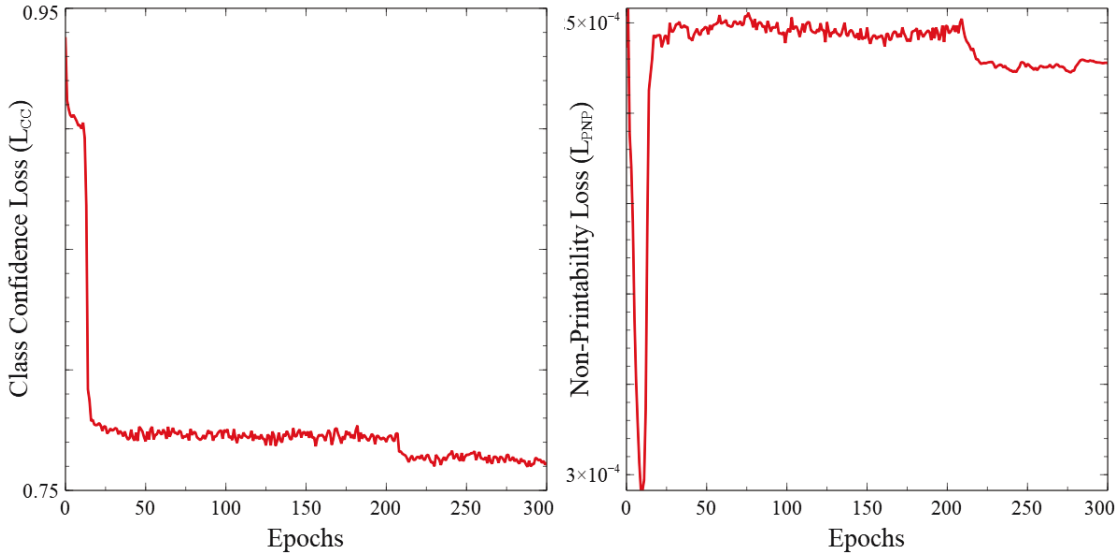
(b) P2-TL

(c) P2-RL

(d) P3-TL

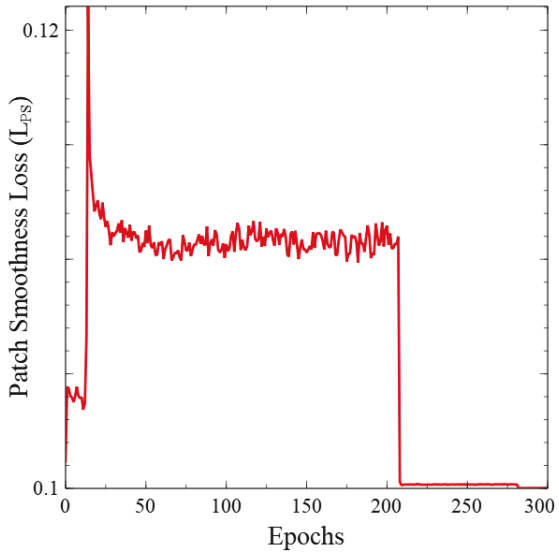
(e) P3-RL

Figure 5.4: First group of adversarial patches (P1, P2, P3) developed in this work to fool a ResNet50-based VMMR system



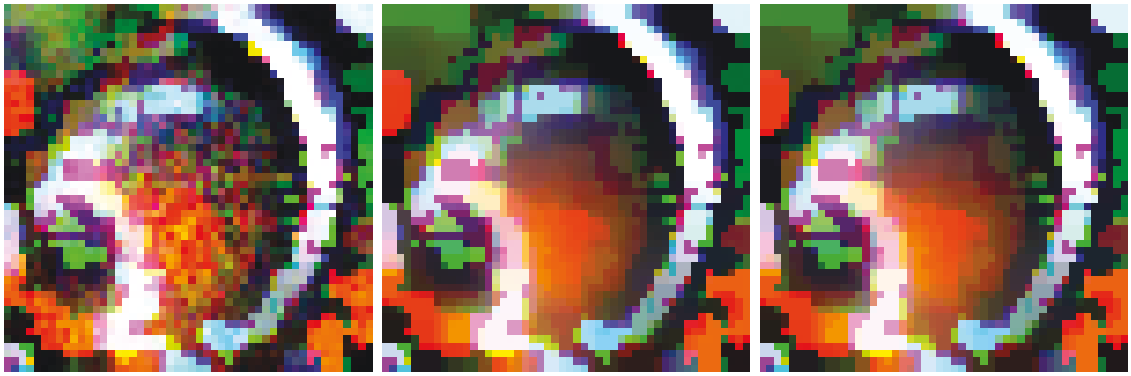
(a) L_{CC}

(b) L_{PNP}



(c) L_{PS}

Figure 5.5: Training of Adversarial Patches P2-RL and P3-RL: Showing the L_{CC}, L_{PNP} and L_{PS} over 300 epochs. P2-RL was extracted from epoch 200 whereas P3-RL was extracted from epoch 300.



(a) P4A (b) P4B (c) P4C

Figure 5.6: Second group of adversarial patches (P4 Series) developed in this work to fool a ResNet50-based VMMR system

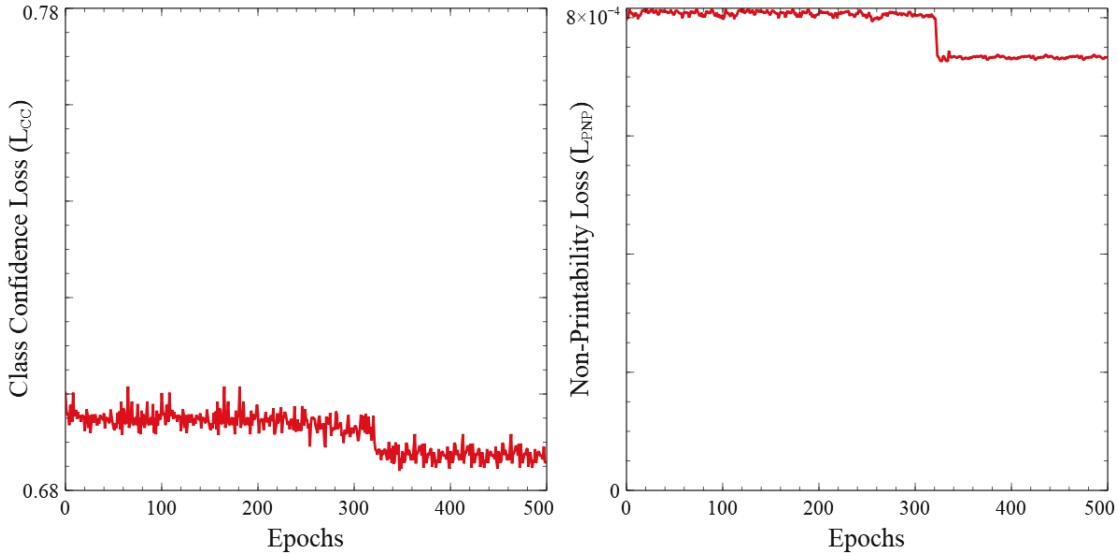
.

placement of patches at fixed (top-left) and random locations. Correspondingly, the suffixes '-TL' and '-RL' to the patch names shall refer to the placement setting in training while the suffixes '(TL)' and '(RL)' refer to the placement settings used in testing.

The patch P1, two versions of P2, and two versions of P3 are shown in Figure 5.4. In this work, we choose a patch size of 50×50 (before transformations) and input images are resized to 224×224. In Figure 5.5, we show the curves for $\mathcal{L}_{CC}$, $\mathcal{L}_{PNP}$, and $\mathcal{L}_{PS}$ obtained during training of P2-RL and P3-RL. While P2-RL was extracted from epoch 200, P3-RL was extracted from epoch 300. The values of all three loss functions are higher at epoch 200 (P2-RL) than at epoch 300 (P3-RL), hinting that P3-RL could be more effective than P2-RL in reducing the VMMR scores.

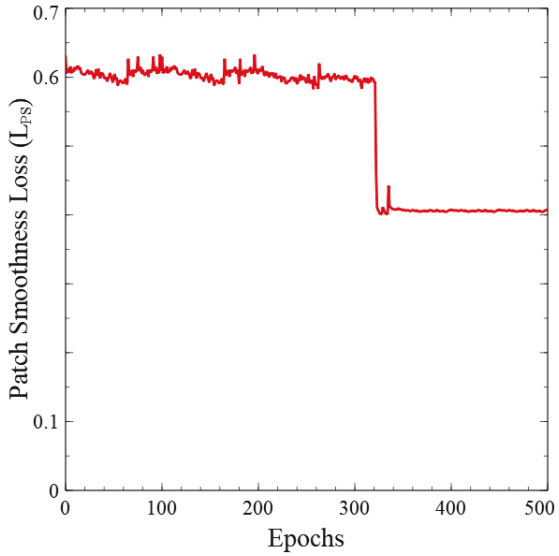
Group 2

The second group of patches we developed are trained with assigning a higher weight to the class confidence loss L_{CC} in Equation 5.3 by setting $\alpha = 10$. A higher weight for L_{CC} would force the learning process to give more importance to reducing the class confidence scores of the target VMMR model than the other components of the loss function. We train three patches in this group, with batch sizes of 256 per epoch and patches placed at



(a) L_{CC}

(b) L_{PNP}



(c) L_{PS}

Figure 5.7: Training of Adversarial Patches P4A/B/C: Showing the L_{CC}, L_{PNP} and L_{PS} over 500 epochs. P4A, P4B and P4C were extracted from epochs 300, 400, and 500, respectively.

random locations in training (RL patch placement setting):

- $P4A$: trained for 300 epochs
- $P4B$: trained for 400 epochs
- $P4C$: trained for 500 epochs.

Figure 5.6 shows the three patches in this group. As one could observe on comparing them with the first group of patches, the P4 patches are less smoother than P1, P2, P3. This is due to the fact that the learning process for P4 (A,B,C) gives lesser importance to the patch smoothness loss L_{PS}. Moreover, there is hardly any visual difference between P4B and P4C, though there are slight changes in the numerical RGB values, hinting that the success of attacks using P4B and P4C may be similar.

Looking at the training curves of P4s (Figure 5.7), one could observe that values of three loss terms (L_{CC}, L_{PNP} and L_{PS}) do not decrease much beyond epoch 400 which explains the close similarity of P4B and P4C besides hinting that training for more epochs may not result in any drastically better adversarial patches.

Group 3

In the third group, adversarial patches trained to fool other object detectors are used as the initial patches in our training process. This can effectively be seen as fine-tuning of adversarial patches (that were pre-trained to attack another application) in order to learn adversarial patches to fool VMMR. In this work, we utilize the patches learnt in [254] as initial patches to learn patches P5A, P5B, and P5C to fool VMMR.

The work of [254] had trained three kinds of adversarial patches, differing in the goal of the optimization process: (a) a patch that is trained to minimize the confidence score of class 'person' as well as objectness score of the targeted model, (b) a patch that is trained to minimize only the objectness score and (c) a patch that is trained to minimize only the confidence score of the 'person' class. The objectness score is more generic than class confidence score and refers to the probability that there exists an object of interest

(regardless of which class it belongs to) in the image at some location. Correspondingly, the three patches we train in Group 3, from these initial patches, are referred to as P5A, P5B, and P5C. Figure 5.8 shows the training curves indicating the loss values obtained during training these patches for 100 epochs. The three resulting patches are shown in Figure 5.9.

5.5 Results & Discussions

In this section, we conduct experiments to investigate the following: (a) performance evaluation of the developed adversarial patches in reducing VMMR scores and performance comparison of these patches, (b) impact of patch placement location on attack effectiveness, and (c) evaluating the effectiveness and robustness of the proposed SIHFR defense method and comparing it against a state-of-the-art defense method.

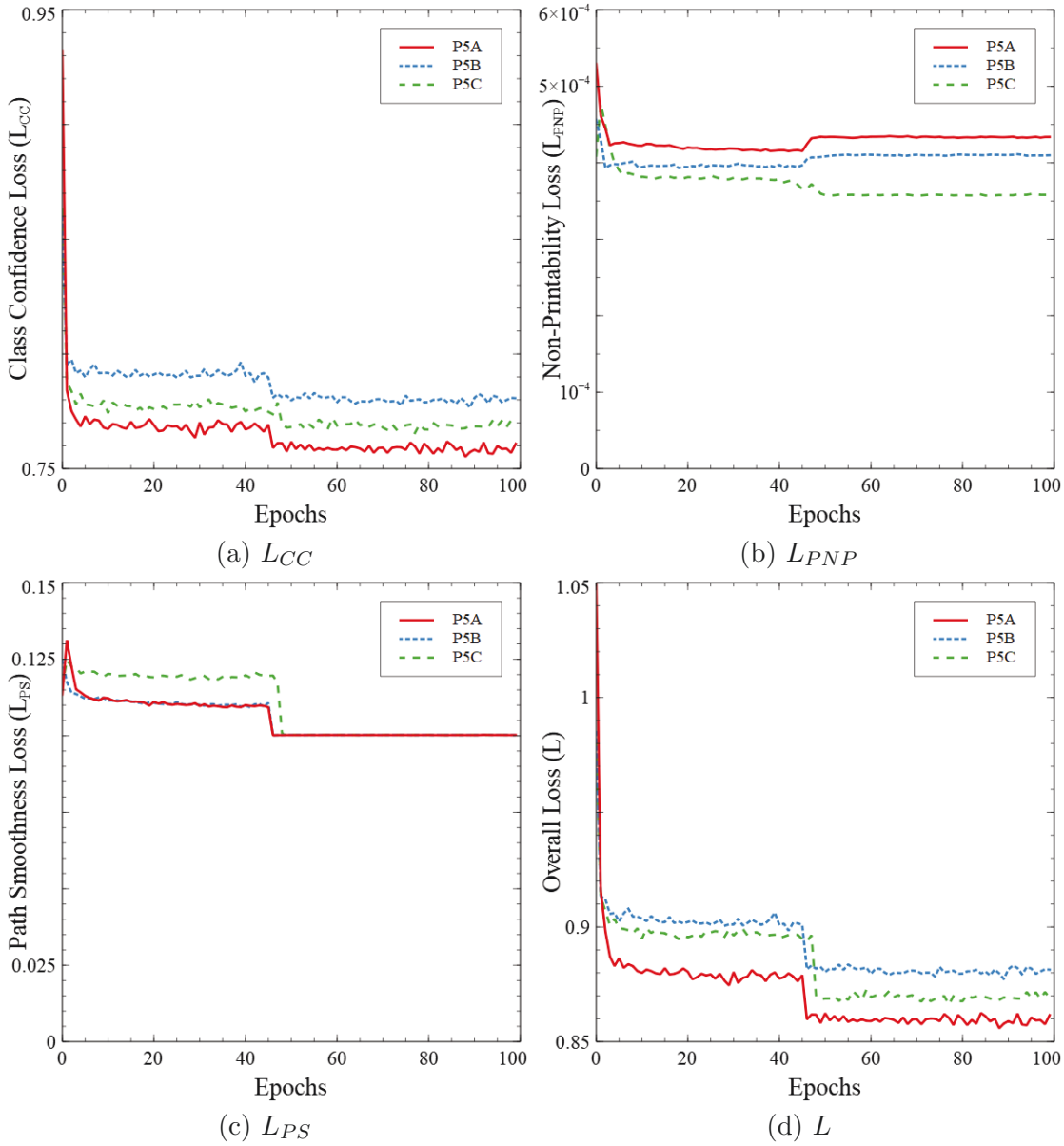
5.5.1 Evaluation of Developed Adversarial Patches

The following subsections present the evaluations of the three groups of adversarial patches developed in this work.

Evaluating Group 1 Patches (P1, P2, P3)

We evaluate P1 under two settings: (a) TL: P1 placed at a fixed location (top-left) on the test images, and (b) RL: P1 placed at random locations on the test images. Table 5.2 summarizes the results of evaluating $P1$ against a ResNet50-based VMMR. The suffix '-TL' to the patch name refers to the top-left placement setting in training while the suffixes '(TL)' and '(RL)' refer to the top-left and random placement settings, respectively, used in testing.

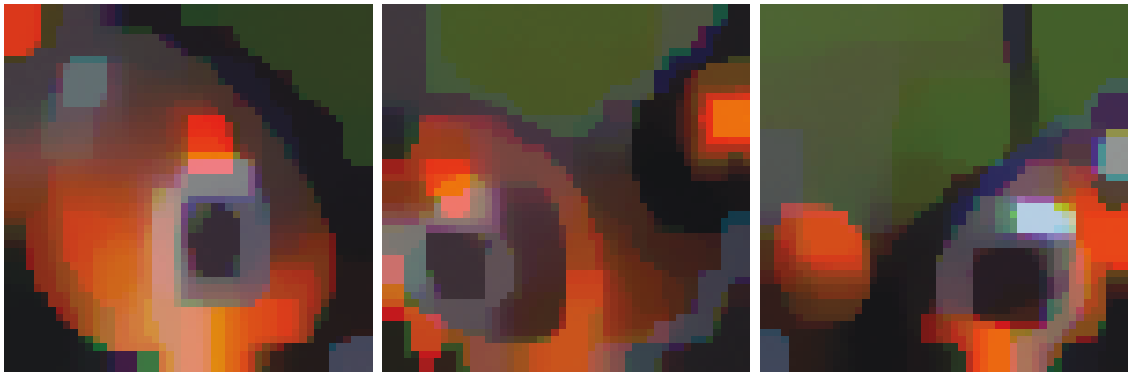
Analyzing the results of Table 5.2, we find that, P1, learnt by fixed placement (at top-left) of training images, achieved the best reduction (though very small) in VMMR precision, recall and F1 scores when it was placed at top-left in test images as well (see



(a) L_{CC} (b) L_{PNP}

(c) L_{PS} (d) L

Figure 5.8: Training of Adversarial Patch P5A/B/C: Showing the L_{CC}, L_{PNP}, L_{PS} and the overall loss L over 100 epochs.



(a) P5A (b) P5B (c) P5C

Figure 5.9: Third group of adversarial patches (P5 Series) developed in this work to fool a ResNet50-based VMMR system, using the patches learnt by [254] as starting points followed by fine-tuning on the VMMR dataset

.

Table 5.2: Evaluation Results of $P1$

Patch	**Avg. Precision**	**Avg. Recall**	**Avg, F1 Score**
None	0.9875	0.9793	0.9821
P1-TL (TL)	0.9780	**0.9639**	**0.9682**
P1-TL (RL)	0.9841	0.9762	0.9790
Random-TL (TL)	0.9872	0.9797	0.9827
Random-TL (RL)	0.9865	0.9780	0.9811

Table 5.3: Evaluation Results of P2-TL

Patch	Avg. Precision	Avg. Recall	Avg, F1 Score
None	0.9875	0.9793	0.9821
P2-TL (TL)	0.9772	0.9699	0.9717
P2-TL (RL)	**0.9733**	**0.9662**	**0.9679**
Random-TL (TL)	0.9854	0.9776	0.9805
Random-TL (RL)	0.9861	0.9787	0.9815

P1-TL (TL) in Table 5.2). The learnt P1 when placed at random locations on test images achieved a very slight reduction in VMMR performance metrics. Moreover, we observe that learnt patches are more effective than random patches in reducing the VMMR system's classification performance.

To evaluate P2, we train two versions of P2, namely: P2-TL and P2-RL. While P2-TL is trained through placement at a fixed location (top-left) on training images, P2-RL is trained through placement at random locations on training images. P2-TL and P2-RL are evaluated under both placement settings TL and RL on test images.

Table 5.3 presents the results of P2-TL-based attacks against the target VMMR system. We note that P2-TL achieved a better reduction in VMMR precision, recall and F1 scores than the random noise patches. When it was placed at random locations (RL placement setting) in test images, P2-TL achieved slightly better reduction than when placed at the top-left location on test images (TL placement setting).

Evaluating P2-RL, we observe the following from Table 5.4. The highest decrease in VMMR performance (precision by 9.72%, recall by 36.15% and F1 score by 29.73%) was achieved by placing P2 at random locations both during training and testing, i.e., by P2-RL (RL). When placed at top-left of test images, P2-RL was able to cause only a slight decrease in the classification scores.

To evaluate P3 as with P2, we train two versions of P3, namely: P3-TL and P3-RL, each evaluated under both TL and RL settings described previously. As we observe the results in Table 5.5, we find that P3-TL causes very slight reduction in precision, recall and F1 scores of VMMR. When placed at random locations on input test images (RL setting),

Table 5.4: Evaluation Results of P2-RL

Patch	Avg. Precision	Avg. Recall	Avg, F1 Score
None	0.9875	0.9793	0.9821
P2-RL (TL)	0.9665	0.9595	0.9595
P2-RL (RL)	**0.8915**	**0.6253**	**0.6901**
Random-RL (TL)	0.9859	0.9780	0.9810
Random-RL (RL)	0.9834	0.9757	0.9786

Table 5.5: Evaluation Results of P3-TL

Patch	Avg. Precision	Avg. Recall	Avg, F1 Score
None	0.9875	0.9793	0.9821
P3-TL (TL)	0.9771	0.9706	0.9722
P3-TL (RL)	**0.9726**	**0.9652**	**0.9668**
Random-TL (TL)	0.9849	0.9783	0.9808
Random-TL (RL)	0.9832	0.9750	0.9781

P3-TL (RL) achieved slightly better reduction in these metrics as compared to that by P3-TL (TL).

Next, we evaluated P3-RL which was trained through placements at random locations on input images. With P3-RL, we note that the best results were obtained under the RL placement settings during testing (see Table 5.6). Most notably, the average recall rate was brought down to 0.6170 from 0.9793, a decrease of around 37.0%.

Evaluating Group 2 Patches (P4A/B/C)

We evaluate P4A, the patch trained for 300 epochs with $\alpha = 10$ as weight to the L_{CC} component of the loss function of Equation 5.3. Table 5.7 summarizes the results. The most effective attack with P4A was under the RL patch placement setting during both training and testing, bringing down the average precision, recall and f1-scores to $0.8836, 0.5242, 0.5978$, respectively. The P4A-RL (RL) achieved 0.4063 points lower average recall than by the

Table 5.6: Evaluation Results of P3-RL

Patch	Avg. Precision	Avg. Recall	Avg, F1 Score
None	0.9875	0.9793	0.9821
P3-RL (TL)	0.9641	0.9566	0.9563
P3-RL (RL)	**0.8974**	**0.6170**	**0.6852**
Random (TL)	0.9855	0.9779	0.9807
Random (RL)	0.9818	0.9738	0.9769

Table 5.7: Evaluation Results of P4A

Patch	Avg. Precision	Avg. Recall	Avg, F1 Score
None	0.9875	0.9793	0.9821
P4A-RL (TL)	0.9498	0.9349	0.9331
P4A-RL (RL)	**0.8836**	**0.5242**	**0.5978**
Random (TL)	0.9860	0.9782	0.9811
Random (RL)	0.9504	0.9305	0.9362

Table 5.8: Evaluation Results of P4B

Patch	Avg. Precision	Avg. Recall	Avg, F1 Score
None	0.9875	0.9793	0.9821
P4B-RL (TL)	0.9469	0.9310	0.9282
P4B-RL (RL)	**0.8843**	**0.5064**	**0.5831**
Random (TL)	0.9853	0.9776	0.9805
Random (RL)	0.9514	0.9335	0.9388

Table 5.9: Evaluation Results of P4C

Patch	Avg. Precision	Avg. Recall	Avg, F1 Score
None	0.9875	0.9793	0.9821
P4C-RL (TL)	0.9472	0.9318	0.9289
P4C-RL (RL)	**0.8826**	**0.5053**	**0.5777**
Random (TL)	0.9859	0.9778	0.9809
Random (RL)	0.9523	0.9332	0.9383

random noise patches under the same patch placement settings (RL) and 0.4551 points lower average recall than the case with no adversarial patches-based attacks.

Next, we evaluate if training P4A for more number of epochs, yielding P4B (at 400 epochs) and P4C (at 500 epochs), could achieve better success in reducing the VMMR scores. Tables 5.8,5.9 summarize the results. The patch P4B, under RL patch placement setting in training and testing, reduced the VMMR average recall score to 0.5064, a 0.4729 points reduction from the no attacks case. The average precision and f1 scores were down by 0.1032 and 0.3990 points, respectively.

The patch P4C was also most effective under the RL patch placement setting in training and testing phases. The average precision, recall and f1 scores were reduced by $0.1049, 0.4740$ and 0.4044 points respectively, in comparison to the VMMR scores in absence of any adversarial patches.

When we compare the performance of P4A, P4B, P4C, we find that P4C achieved 0.0011 and 0.0189 points lower average recall score than P4B and P4A, respectively. The gains of training P4C for 100 more epochs than P4B were minimal.

With all three versions of P4, we find that the patch placement at top-left of the input images during testing achieved minimal success in lowering the VMMR scores in comparison to the success achieved by the patches under random location (RL) patch placement strategy. While P4A-RL (TL) lowered the average recall by 0.0444 points, P4B-RL (TL) and P4C-RL (TL) lowered it by 0.0483 and 0.0475 points respectively, when compared to the scores obtained in absence of any adversarial patches. In contrast, these patches achieved around $10\times$ more reduction under the random location patch placement

strategy in testing (denoted by the patch name and suffix '(RL)' in the respective Tables).

Evaluating Group 3 Patches (P5A/B/C)

There are three versions of P5, as described in 5.4.3: P5A, P5B, and P5C. Tables 5.10,5.11, and 5.12 summarize the evaluations results respectively. The three patches were fine-tuned (starting from the three patches of [254]) and trained by randomly placing the patches on training images (i.e., under the RL patch placement setting).

On the testing dataset, the patch P5A achieved a reduction in average recall scores by 0.0229 points under the TL patch placement strategy and by 0.3786 points under the RL patch placement strategy, the latter achieving 37.2% more success than the former in reducing the average recall score.

Looking at the performance of P5B, we find that it reduced the average precision, recall and f1 scores of VMMR by 0.0751, 0.3031, and 0.2388 points respectively, following the RL patch placement strategy. The reduction in average recall achieved by P5B-RL (TL), i.e. with TL patch placement strategy on testing images, was only by 0.0146 points.

The patch P5C achieved best results when placed at random locations on test images, i.e., following the RL patch placements strategy in testing. It achieved a reduction in average precision, recall and f1 scores by 0.0796, 0.2660 and 0.2135 points respectively when compared against the case of no adversarial patches-based attacks. With P5C as well, following the TL patch placement strategy on testing images yielded minimal success in reducing the VMMR scores. The average recall score reduction achieved by P5C-RL (TL) was only by 0.0180 points.

Upon comparing P5A, P5B and P5C based on their best results in Tables 5.10,5.11, and 5.12, we make the following observations. The patch P5A was most successful in reducing the VMMR scores, followed by P5B and P5C. P5A yielded 38.7% reduction in average recall scores, whereas P5B and P5C achieved 31% and 27.2% reduction from the average recall obtained under no adversarial patches-based attacks. This indicates that the patch of [254] that was trained to minimize class confidence score as well as objectness score serves as a better initial patch for fine-tuning than their other two patches which minimized either the class confidence score or the objectness score alone. These three patches were used in

Table 5.10: Evaluation Results of P5A

Patch	Avg. Precision	Avg. Recall	Avg, F1 Score
None	0.9875	0.9793	0.9821
P5A-RL (TL)	0.9655	0.9564	0.9571
P5A-RL (RL)	**0.9057**	**0.6007**	**0.6770**
Random (TL)	0.9863	0.9784	0.9814
Random (RL)	0.9521	0.9324	0.9378

Table 5.11: Evaluation Results of P5B

Patch	Avg. Precision	Avg. Recall	Avg, F1 Score
None	0.9875	0.9793	0.9821
P5B-RL (TL)	0.9728	0.9647	0.9662
P5B-RL (RL)	**0.9124**	**0.6762**	**0.7433**
Random (TL)	0.9855	0.9779	0.9807
Random (RL)	0.9537	0.9330	0.9389

the fine-tuning process to obtain patches P5A, P5B and P5C respectively, as described in Section 5.4.3.

From the above evaluations, we observe the following. First, trained adversarial patches are more effective than random noise patches in reducing the classification performance of VMMR. Second, assigning a higher weight to the L_{CC} component of the loss function of Equation 5.3 results in adversarial patches that are not as smooth as patches learnt with equally weighted components of the loss function. For example, the patches P4A,B,C are less smoother than patches P1,P2, or P3 (compare Figures 5.4 and 5.6). Third, the trained adversarial patches are more effective in fooling the VMMR system when placed at random locations on the test images, i.e. under the RL patch placement strategy, than when placed at the top-left location on test images.

Table 5.12: Evaluation Results of P5C

Patch	Avg. Precision	Avg. Recall	Avg, F1 Score
None	0.9875	0.9793	0.9821
P5C-RL (TL)	0.9708	0.9613	0.9627
P5C-RL (RL)	**0.9079**	**0.7133**	**0.7686**
Random (TL)	0.9862	0.9780	0.9811
Random (RL)	0.9440	0.9301	0.9334

5.5.2 Comparison of Developed Adversarial Patches

To compare the effectiveness of the developed patches we choose to look at the reduction in average recall score of the targeted VMMR system as it most representatively reflects the success of an adversarial attack [254]. In Table 5.13, we present the reduction in recall scores achieved by the best settings for each patch, compared against the baseline average recall of VMMR when no adversarial patches are present.

The higher reduction in recall caused by P3-TL (RL) vs. P2-TL (RL) and P3-RL (RL) vs. P2-RL (RL) indicates, as expected, that an adversarial patch trained for more number of epochs is more effective, given the same strategy of placing the patches. Amongst the patches P1, P2 and P3, the best reduction in recall, of almost 37.0%, is achieved by P3-RL (RL), indicating that placing the adversarial patches at random locations on images during training and execution is most effective in fooling the ResNet50-based VMMR system.

Amongst the patches of all three groups, we find that the best results were achieved by P4C-RL (RL) which reduced the average recall score by 48.40%. On the other hand, amongst the patches tested with the TL patch placement strategy, P4B-RL (TL) achieved the best reduction in recall score (by 4.93%).

To answer the question "Whether adversarial patches learnt from fine-tuning of patches pre-trained for another application are more effective than those learnt from scratch targeting the VMMR application?", we compare the results of the patches P4 and P5. The best reduction in VMMR scores achieved by P5 was with P5A-RL (RL), bringing down the average precision, recall and f1 scores by 8.28%, 38.66% and 31.07% respectively. In contrast, the best reduction in scores achieved by P4 was by 10.62%, 48.40%, and 41.18%

Table 5.13: Comparing Effectiveness of the Developed Adversarial Patches

Patch/Setting	**Avg. Recall**	**Reduction (%)**
No Patch	0.9793	-
P1-TL (TL)	*0.9639*	*1.57*
P2-TL (RL)	0.9662	1.34
P2-RL (RL)	0.8891	9.21
P3-TL (RL)	0.9652	1.44
P3-RL (RL)	0.6170	37.0
P4A-RL (TL)	*0.9349*	*4.53*
P4A-RL (RL)	0.5242	46.47
P4B-RL (TL)	***0.9310***	***4.93***
P4B-RL (RL)	0.5064	48.29
P4C-RL (TL)	*0.9318*	*4.85*
P4C-RL (RL)	**0.5053**	**48.40**
P5A-RL (TL)	*0.9564*	*2.34*
P5A-RL (RL)	0.6007	38.66
P5B-RL (TL)	*0.9647*	*1.49*
P5B-RL (RL)	0.6762	30.95
P5C-RL (TL)	*0.9613*	*1.84*
P5C-RL (RL)	0.7133	27.16

respectively, with P4C-RL (RL). So, P4 achieved 2.34, 9.74, 10.11 percentage points higher reduction in the average VMMR precision, recall and f1 scores respectively, in comparison to that by P5. This indicates that training adversarial patches from scratch could be more effective against ResNet50-based VMMR models than fine-tuning patches pre-trained to attack models trained for other applications such as person detection. In addition, it indicates that a higher weight to the $\mathcal{L}_{CC}$ during training (e.g., of P4) yields more effective adversarial patches.

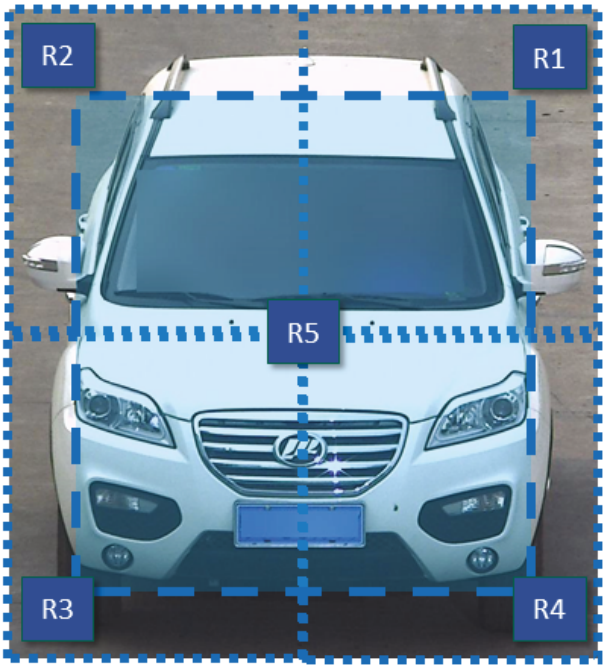


Figure 5.10: Demarcating the five regions used in Section 5.5.3 to investigate the impact of patch placement location on effectiveness of adversarial attacks against the target VMMR system, using the CompCars dataset [265].

5.5.3 Impact of Patch Placement Location on Attack Effectiveness

Based on the results from the previous experiments, we learnt that placing the patches at random locations on the input images, both during training and testing, yields the most effective attacks. Next, we investigate which regions of the input image are more effective than others for the random patch placement.

We study five regions in these experiments, as depicted in Figure 5.10: (i) R1, the top-right quadrant, (ii) R2, the top-left quadrant of the input image, (iii) R3, the bottom-left quadrant, (iv) R4, the bottom-right quadrant, and (v) R5, a central region on the input image focusing on the vehicle's face. In each region, the patches are randomly placed on the input images, constrained by the region boundaries. For example, patches to be placed in R1 cannot be placed on any location in the top-left quadrant (R2). In these experiments, we utilize the P4C-RL patch which achieved the best results in the previous experiments.

Given a patch size of 50×50 and input image size of 224×224, the patch width (height) is around 22% of the image width (height). So, the patches in each of the five regions have their center coordinates (x, y) restricted by certain factors of the input image width and height, as presented in Table 5.14, to ensure that patches are contained within their intended quadrants. For example, a patch in R1 could have a minimum (x, y) coordinate of $(0.5 + W_P/W_I, 0.0 + W_P/W_I)$ and a maximum (x, y) coordinate of $(1.0 - W_P/W_I, 0.5 - W_P/W_I)$, where W and H represent width and height, while subscripts I and P refer to input images and patches respectively. In Figure 5.11, we show some samples of the vehicle images attacked with adversarial patches placed randomly within each of the regions R1-R5 respectively.

In Table 5.15, we present the evaluation results of placing the adversarial patch in each of the five regions. The lowest reduction in average recall score for VMMR was obtained when the patch was placed in R4 while placement of the adversarial patch in R1 yielded the second lowest reduction in average recall score. This indicates that placement of adversarial patches in R1 and R4 is not the most effective approach of an attack. A reason for this could be that these regions of a vehicle's image are not the most informative or important ones in a CNN trained for VMMR, possibly due to limited discriminatory

Figure 5.11: Sample vehicle images attacked with adversarial patches placed randomly within regions R1-R5 respectively (images from CompCars dataset [265]).

Table 5.14: Restrictions on Adversarial Patch Coordinates in the Five Regions

Region	**Min.** x	**Min.** y	**Max.** x	**Max.** y
R1	$0.5 + W_P/W_I$	$0.0 + W_P/W_I$	$1.0 - W_P/W_I$	$0.5 - W_P/W_I$
R2	$0.0 + W_P/W_I$	$0.0 + W_P/W_I$	$0.5 - W_P/W_I$	$0.5 - W_P/W_I$
R3	$0.0 + W_P/W_I$	$0.5 + W_P/W_I$	$0.5 - W_P/W_I$	$1 - W_P/W_I$
R4	$0.5 + W_P/W_I$	$0.5 + W_P/W_I$	$1.0 - W_P/W_I$	$1.0 - W_P/W_I$
R5	$0.2 + W_P/W_I$	$0.2 + W_P/W_I$	$0.8 - W_P/W_I$	$0.8 - W_P/W_I$

Table 5.15: Evaluating the Impact of Patch Placement Location on P4C's Attack Effectiveness

Region	**Avg. Precision**	**Avg. Recall**	**Avg. F1 Score**
No Patch	0.9875	0.9793	0.9821
R1	0.9343	0.8567	0.8732
R2	0.9228	0.8331	0.8485
R3	0.9058	0.7971	0.8181
R4	0.9293	0.8887	0.8939
R5	**0.8821**	**0.4871**	**0.5521**

features occurring in R1.

From Table 5.15, we find that the most effective region to place the adversarial patch P4C was R5, achieving a reduction of 10.67%, 50.26% and 43.78% in the average precision, recall and f1 scores of the VMMR system. Similarly, the adversarial patch P5A also performed most effective in region R5, achieving reductions of 8.02%, 33.87% and 27.38% in the average scores respectively. Although R5 overlaps with R1, R2, R3 and R4, it tightly encloses a vehicle's face unlike the other quadrants that include regions away from the vehicle's face as well. This could be the reason why patches placed randomly within R5 had the most impact in reducing VMMR scores.

Table 5.16: Evaluating the Impact of Patch Placement Location on P5A's Attack Effectiveness

Region	Avg. Precision	Avg. Recall	Avg. F1 Score
No Patch	0.9875	0.9793	0.9821
R1	0.9603	0.9370	0.9413
R2	0.9510	0.9183	0.9242
R3	0.9383	0.8869	0.8987
R4	0.9457	0.9008	0.9115
R5	**0.9083**	**0.6476**	**0.7132**

5.5.4 Evaluating the Proposed SIHFR Defense Method

We evaluate the effectiveness of the proposed SIHFR defense method against adversarial patches trained to fool CNN-based VMMR systems. Moreover, we analyse the overhead incurred by he addition of SIHFR module to the VMMR module. We also compare the performance of SIHFR with that of a related state-of-the-art defense method in defending the VMMR system from adversarial patches. In addition, we study the effect of patch size on the robustness of the proposed SIHFR defense method.

Defending against the Adversarial Patches

In this section, we study robustness of the proposed defense method against the developed adversarial patches. Based on previous experiments (without any defense method in place), the two best adversarial patches were P4C and P5A. The most effective patch placement region was found to be R5. We test the proposed defense method's success in eliminating or reducing the impact of adversarial patches P4C and P5A on the target VMMR system. The patches are placed randomly within the R5 region in separate experiments. Table 5.17 shows the average recall scores (at 95% confidence intervals) of the target VMMR system with and without SIHFR in the presence and absence of adversarial patches.

The target VMMR system with SIHFR activated experienced a slight decrease in average recall scores when there were no adversarial patches. This is because the target VMMR

Table 5.17: Evaluating SIHFR against P4C- and P5A-based Attacks

Patch/Region	**Avg. Recall (no SIHFR)**	**Avg. Recall (with SIHFR)**	**Avg. Improvement (%)**
No Patch Attack	0.9862 ± 0.01	0.9657 ± 0.01	-2.1
P4C-R5	0.4941 ± 0.05	0.8365 ± 0.01	69.28
P5A-R5	0.6719 ± 0.04	0.8080 ± 0.02	20.25

system which is based on ResNet50 model was not trained using benign images constructed using the symmetric image halves as in SIHFR, but was trained on un-modified images.

On the other hand, the average recall scores of the target VMMR system improved with our proposed SIHFR defense method by 69.28% and 20.25% on average, in the case of P4C- and P5A-based attacks respectively. These results indicate the effectiveness of the proposed SIHFR method in eliminating the influence of adversarial patches on the VMMR system to a significant extent, though there still remains room for further improvement.

Evaluating SIHFR's Overhead on the target VMMR System

In order to evaluate the overhead added to a CNN-based VMMR system by activating the proposed SIHFR defense method, we examine the processing time consumed by the SIHFR module in comparison to the processing time of the CNN model itself (without the SIHFR defense method).

The average processing time (per image), at a 95% confidence interval, taken by the SIHFR method was 1.65 ± 0.22ms whereas that consumed by the ResNet50-based VMMR model (without SIHFR) was around 5.33 ± 0.08ms. Activating the SIHFR module slightly increased the average VMMR processing time to 6.98 ± 0.30ms, adding an overhead of $27.33\% - 34.62\%$ which amounts to less than 2ms per image (on average). It is also worth mentioning that SIHFR does not require re-training of the CNN-based VMMR model and can work as a connected complementary module, potentially serving as a lightweight trigger to launch more advanced or sophisticated defenses.

Comparative Study

This thesis focuses on developing a lightweight defense method that pre-processes input images to eliminate or reduce the influence of adversarial patches. In doing so, two main design goals that need to be satisfied are: (i) the defense method should not require re-training or modifications of the CNN-based VMMR model, and (ii) the processing time required by the defense method, per image, should be minimal. From the few defense methods that have been proposed in the literature to mitigate the effect of adversarial patches through input image pre-processing, the recent work of [1] is closest in spirit to our work.

The major factor we examine when comparing SIHFR with [1]'s Ally-Patches is the average processing time consumed by the methods per input image. Using a common computing platform, without any hardware accelerators such as GPUs, we run both methods on test images attacked with adversarial patches. While the SIHFR method designed and developed in this thesis consumed on average, $2.60 \pm 0.13ms$ (at a 95% confidence interval), the Ally-Patches method required $922.26 \pm 32.78ms$ on average. The slow performance of Ally-Patches is due to the sliding window-based approach to extract candidate ally patches followed by a procedure to filter out some patches which is based on overlap criteria or mutual information [221] constraints.

It is worth mentioning that the method proposed in [1] requires the CNN-based detection models to be trained to detect or classify objects based on incomplete and partial views (patches) of objects of interest. This requirement puts an additional limitation for CNN-based VMMR models due to the multiplicity and ambiguity challenges described in Section 2.3.3.

Effect of Patch Size on Defense Robustness

In these set of experiments, we investigate the robustness of SIHFR defense method against adversarial patches placed at five different scales with the R5 region on the image. The five scales for the patches used in these experiments are: 25×25, 40×40, 50×50, 60×60, and 70×70. We choose to conduct the evaluations with the two best patches developed: P4C and P5A.

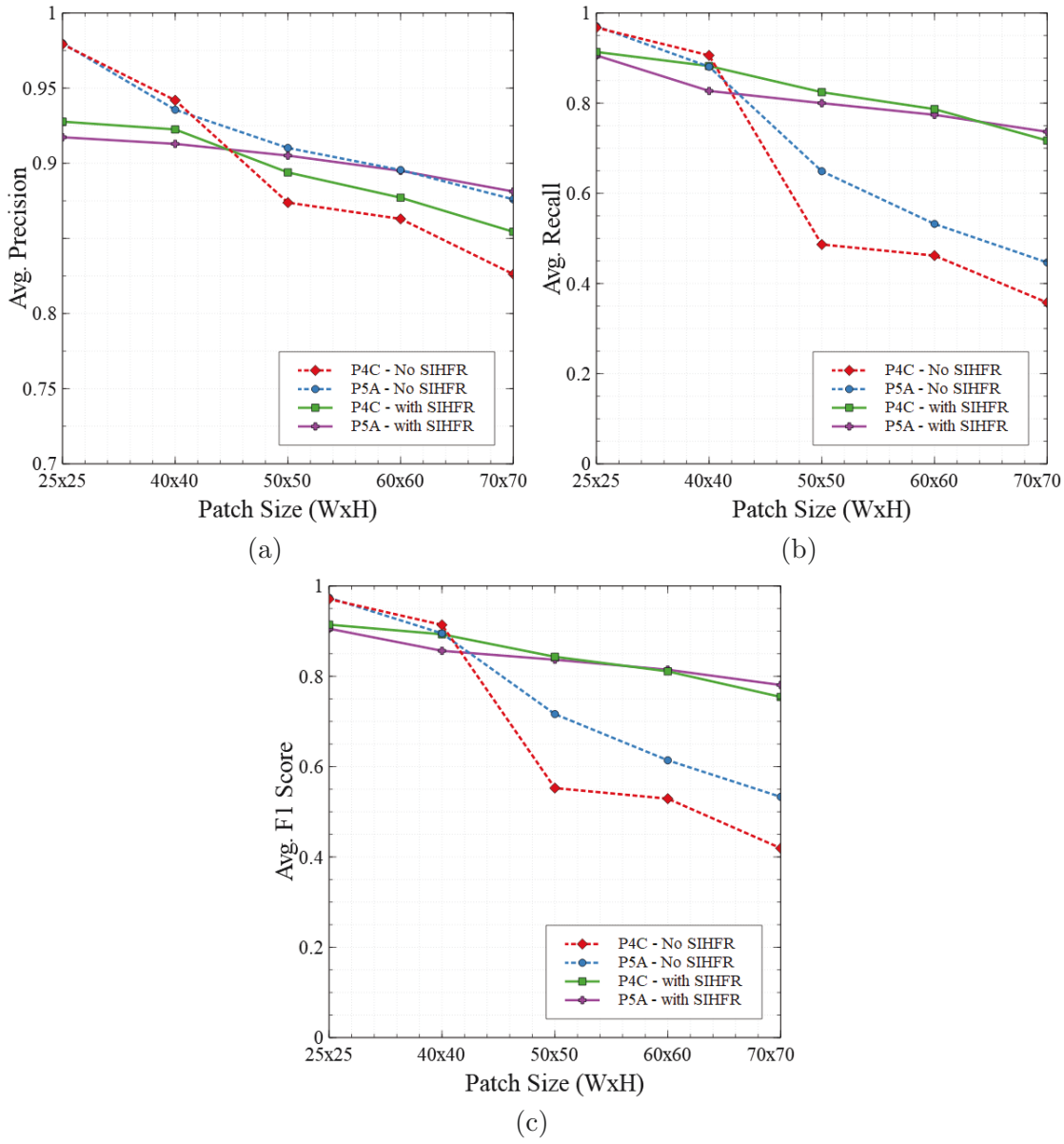
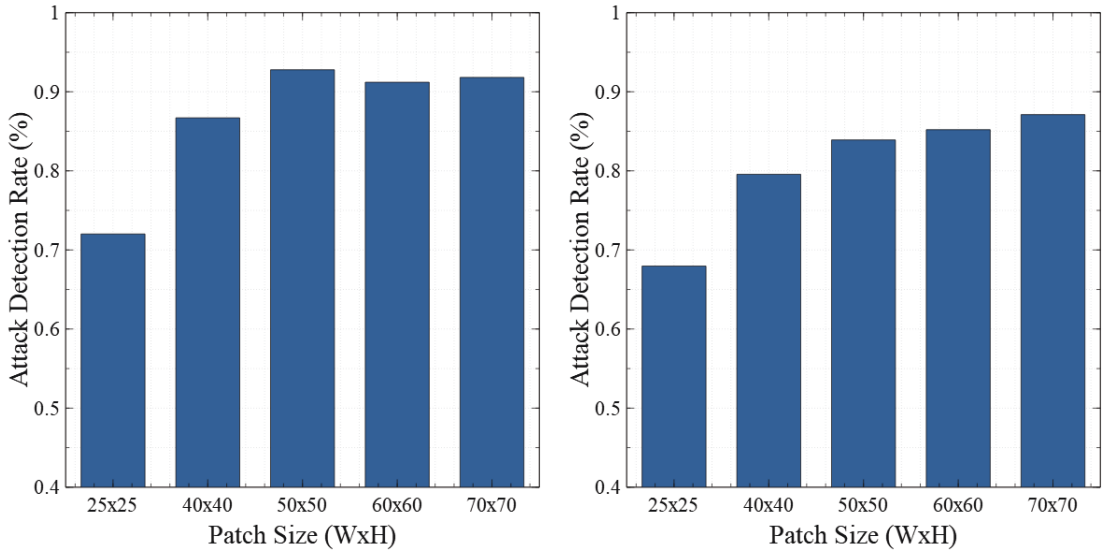


Figure 5.12: Effect of patch size on SIHFR's robustness, measured in terms of improvements in VMMR performance: (a) Average Precision, (b) Average Recall, and (c) Average F1 scores. The attacks were launched using P4C and P5A. Higher the score, in comparison to the case with no SIHFR defense, the higher is SIHFR's robustness.



(a) P4C (b) P5A

Figure 5.13: Comparing SIHFR's attack detection rate at different sizes of adversarial patches P4C and P5A.

In Figure 5.12, we provide the results in terms of average precision, recall and F1 scores, with and without the SIHFR defense method against attacks launched using different sizes of P4C and P5A. As one may observe, with adversarial patches of sizes above 40×40, SIHFR improves the performance scores of VMMR. This demonstrates the effectiveness of SIHFR in improving the robustness of the CNN-based VMMR model. Moreover, we find that the SIHFR method is highly robust against adversarial patches of sizes as low as 50×50 and as large as 70×70, given input images of sizes 224×224.

With small adversarial patches, e.g. of size 25×25, though their impact on VMMR performance is limited, the benefits of activating SIHFR defense method are not that evident. However, looking at the attack detection rates of SIHFR (in Figure 5.13), we note that SIHFR still succeeded in detecting around 72% of attacks launched using P4C and around 68% of attacks launched using P5A, at a patch size of 25×25.

5.6 Concluding Remarks

In this chapter, we studied and introduced adversarial patches that could be placed on vehicles to fool or circumvent automated vehicle surveillance systems such as VMMR. Through experimental evaluations on a real-world surveillance nature dataset of vehicles, we demonstrate the effectiveness of the developed adversarial patches in reducing the classification performance of a robust VMMR system that is based on a popular convolutional neural network model known as ResNet50. A reduction of upto 48% in recall score was achieved by the developed patches. In addition, we evaluated two patch placement strategies: fixed (at top-left) vs. random, and found that adversarial patches placed at random locations in the image during training and execution are more effective. Moreover, we designed and developed a lightweight defense method called SIHFR to eliminate the effect of adversarial patches-based attacks on VMMR systems by making use of symmetry in vehicle images. The proposed defense method succeeded in defending against 83.40% to 92.15% of the attacks (with patches of size 50×50), leading to an improvement in VMMR performance by up to 69.28%.

To the best of our knowledge, this is the first work that investigates the problem of adversarial learning against VMMR systems and proposes a lightweight defense method for the same. It is hoped that this work shall pave the path forward for more studies in developing VMMR systems that are highly robust to adversarial learning-based attacks, in a quest to achieve more secure and adversarially robust smart city surveillance systems. In future, we plan to develop a defense method against multi-adversarial patches-based attacks that target VMMR systems.

Chapter 6

Conclusion and Future Work

We present the conclusion and summary of this thesis in Section 6.1, followed by outlining interesting directions of future research in Section 6.2.

6.1 Thesis Summary

The work in this thesis proposed and investigated solutions to enhance the security of internet of things, especially the connected surveillance systems, against known and unknown network intrusion attacks (at the cyber network-level) and adversarial patches-based attacks (at the physical world-level).

In order to detect known network intrusions, at the cyber network-level, through a supervised learning-based methodology, this thesis designed and developed *TempoCode*, a temporal codebook-based encoding of flow features, as a novel feature transformation of network flow features based on capturing the key patterns of benign traffic in a learnt temporal codebook. Using the codebook, distances of (benign and malicious) traffic flows from those key patterns are measured and recorded as the transformed features to train an ensemble of SVM classifiers. The experimental evaluations on recent realistic datasets (CICIDS2017 and NBaIoT) proved the effectiveness of TempoCode representations in detecting intrusions of various types and for different devices used in the datasets, obtaining high precision scores, low false positive rates, and low detection times on average.

To address the problem of detecting unknown network intrusion attacks, this thesis designed an unsupervised learning-based methodology leveraging neural networks-based autoencoders (AEs). Specifically, this thesis developed, evaluated and compared four methods to build efficient and adaptive ensembles of AEs for intrusion detection in internet of things, focusing on connected surveillance cameras. In doing so, the proposed methods orchestrate and conduct de-activations of constituent AEs in the ensemble. These methods differ in how the candidate AEs are selected for de-activation: Criteria-based De-Activation (CDA) and Random De-Activation (RDA). Moreover, the proposed methods differ in when the de-activations are performed: In-Training De-activation (ITD) and Post-Training De-activation (PTD). Through comprehensive experiments on realistic IoT datasets, we showed that the proposed methods enable achieving a less costly ensemble of AEs (in terms of computation, training, re-training and inference time costs) at satisfactory levels of detection performance (in terms of AUC and EER scores). In massive IoT environments where an IDS is composed of ensembles of AEs for each device or device-type, the methods developed in this thesis provide for more efficient ensembles which could be scaled up or down adaptively, paving the path towards a scalable and efficient intrusion detection system or service that could be deployed on-device or on-edge.

Addressing the security issues at physical-world level, this thesis first developed, studied and examined adversarial patches that could be placed on vehicles to fool or circumvent automated vehicle surveillance systems such as VMMR. Through experimental evaluations on a real-world surveillance nature dataset of vehicles, this thesis demonstrated the effectiveness of the developed adversarial patches in reducing the classification performance of a robust VMMR system that is based on a popular convolutional neural network model known as ResNet50. A reduction of upto 48% in recall score was achieved by the proposed patches. In addition, this thesis investigated two patch placement strategies: fixed (at top-left) vs. random, and found that adversarial patches placed at random locations in the image during training and execution were more effective. Moreover, this thesis designed and developed a lightweight defense method, termed as *SIHFR*, to eliminate the effect of adversarial patches-based attacks on VMMR systems by making use of symmetry in vehicle images. The proposed defense method succeeded in defending against 83.40% to 92.15% of the attacks, leading to an improvement in VMMR performance by 22.31% to 68.24%. To the best of our knowledge at the time of this thesis, this is the first work that investigated

the problem of adversarial learning against VMMR systems and proposed a lightweight defense method for the same. It is strongly believed that this work shall guide and pave the path forward for further studies in developing VMMR systems that are highly robust to adversarial learning-based attacks.

6.2 Future Research Directions

In this section, we briefly discuss some additional issues and challenges of interest associated with securing the evolving connected and automated surveillance systems, against network intrusions and adversarial patches-based physical world attacks. Although considerable efforts have been made in the field of cybersecurity, achieving fully robust, secure and resilient heterogeneous IoT environments such as those involving connected and automated surveillance systems is an ongoing pursuit. The following aspects are planned to be investigated as future work:

- Real Large-scale Datasets: In order to foster research in developing IDSs for heterogeneous IoT, efforts are needed to build publicly available large-scale datasets using real devices and realistic scenarios. Currently, most works use simulations or small-scale datasets.
- Interpretability: Further studies are needed to derive insights on the reasoning behind a method's good detection performance for some attacks and not for some other attacks, potentially through machine learning interpretability studies [183].
- Heterogeneous Internet of Cameras and Things: In addition, further work is planned to investigate enhancing the performance of this thesis' proposed methods on heterogeneous cameras and other IoT devices, in terms of computation, storage and inference-time costs. With the diversity of attacks and devices, while some features may be important to identify certain kinds of malicious anomalies, some features may not be. It remains a challenge to adaptively assess feature importance to enable dynamic feature selection in live IDSs.

- Adversarial Machine Learning-based Attacks: An issue with neural networks-based solutions is the vulnerability to adversarial machine learning attacks [142, 148] that could learn to fool or evade the IDS. Hence, an important future work in plan is to design and develop adversarially robust IDSs and services.

- Multi-Adversarial Patches-based Attacks: The focus of this thesis was on attacks launched against CNN-based VMMR systems using single adversarial patches only. Further investigations are intended to assess the potential impact of multiple adversarial patches per vehicle image on VMMR performance. In light of this, it is planned to design and develop more sophisticated defense methods to detect and mitigate the effect of multiple adversarial patches-based attacks.

While different methods have been proposed to detect anomalous malicious traffic, raising the bar for successful intrusions attacks, achieving a fully robust IDS that is always accurate with zero false alarm rates and incurs minimal overhead, is still an ongoing pursuit. In the context of VMMR surveillance systems, there are other kinds of adversarial attacks that could be performed to circumvent detection or tracking. For example, adversaries could customize vehicles in different ways, e.g., adding accessories such as spoilers, neon lights or other fixtures that do not usually appear on vehicles of that make and model, effectively modifying the original appearance. Designing vision-based surveillance systems that are robust to such cases needs further attention.

References

[1] Alaa E. Abdel-Hakim. Ally patches for spoliation of adversarial patches. *J. Big Data*, 6:51, 2019.

[2] M. AbdelMaseeh, I. Badreldin, M.F. Abdelkader, and M. El Saban. Car Make and Model recognition combining global and local cues. In *21st International Conference on Pattern Recognition (ICPR)*, pages 910–913, Nov 2012.

[3] Kaouther Abrougui, Azzedine Boukerche, and Richard Werner Nelem Pazzi. Design and evaluation of context-aware and location-based service discovery protocols for vehicular networks. *IEEE Trans. Intell. Transp. Syst.*, 12(3):717–735, 2011.

[4] O. Abumansoor and A. Boukerche. A secure cooperative approach for nonline-of-sight location verification in vanet. *Vehicular Technology, IEEE Transactions on*, 61(1):275–285, Jan 2012.

[5] Elie El Ajaltouni, Azzedine Boukerche, and Ming Zhang. An efficient dynamic load balancing scheme for distributed simulations on a grid infrastructure. In David J. Roberts, Abdulmotaleb El-Saddik, and Alois Ferscha, editors, *12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications, 27-29 October 2008, Vancouver, BC, Canada, Proceedings*, pages 61–68. IEEE Computer Society, 2008.

[6] Naveed Akhtar, Jian Liu, and Ajmal Mian. Defense against universal adversarial perturbations. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 3389–3398. IEEE Computer Society, 2018.

[7] Tamer Aldwairi, Dilina Perera, and Mark A. Novotny. An evaluation of the performance of restricted boltzmann machines as a model for anomaly network intrusion detection. *Computer Networks*, 144:111 – 119, 2018.

[8] Noura Aljeri, Kaouther Abrougui, Mohammed Almulla, and Azzedine Boukerche. A performance evaluation of load balancing and qos-aware gateway discovery protocol for vanets. In Leonard Barolli, Fatos Xhafa, Makoto Takizawa, Tomoya Enokido, and Hui-Huang Hsu, editors, *27th International Conference on Advanced Information Networking and Applications Workshops, WAINA 2013, Barcelona, Spain, March 25-28, 2013*, pages 90–94. IEEE Computer Society, 2013.

[9] Noura Aljeri, Kaouther Abrougui, Mohammed Almulla, and Azzedine Boukerche. A reliable quality of service aware fault tolerant gateway discovery protocol for vehicular networks. *Wirel. Commun. Mob. Comput.*, 15(10):1485–1495, 2015.

[10] Noura Aljeri, Mohammed Almulla, and Azzedine Boukerche. An efficient fault detection and diagnosis protocolfor vehicular networks. In Mirela Sechi Moretti Annoni Notare and Abdelhamid Mammeri, editors, *Proceedings of the third ACM international symposium on Design and analysis of intelligent vehicular networks and applications, DIVANet@MSWiM 2013, Barcelona, Spain, November 3-4, 2013*, pages 23–30. ACM, 2013.

[11] Noura Aljeri and Azzedine Boukerche. Performance evaluation of movement prediction techniques for vehicular networks. In *IEEE International Conference on Communications, ICC 2017, Paris, France, May 21-25, 2017*, pages 1–6. IEEE, 2017.

[12] Noura Aljeri and Azzedine Boukerche. A predictive collision detection protocol using vehicular networks. In *28th IEEE Annual International Symposium on Personal, Indoor, and Mobile Radio Communications, PIMRC 2017, Montreal, QC, Canada, October 8-13, 2017*, pages 1–5. IEEE, 2017.

[13] Noura Aljeri and Azzedine Boukerche. An efficient movement-based handover prediction scheme for hierarchical mobile ipv6 in vanets. In Mónica Aguilar-Igartua, Carolina Tripp Barba, and Ahmad Mohamad Mezher, editors, *Proceedings of the 15th*

ACM International Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks, PE-WASUN@MSWiM 2018, Montreal, QC, Canada, October 28 - November 02, 2018, pages 47–54. ACM, 2018.

[14] Noura Aljeri and Azzedine Boukerche. Mobility and handoff management in connected vehicular networks. In Wessam Ajib, Rodolfo W. L. Coutinho, and Frank Li, editors, *Proceedings of the 16th ACM International Symposium on Mobility Management and Wireless Access, MobiWac 2018, Montreal, QC, Canada, October 28 - November 02, 2018*, pages 82–88. ACM, 2018.

[15] Noura Aljeri and Azzedine Boukerche. An efficient handover trigger scheme for vehicular networks using recurrent neural networks. In Geyong Min and Ahmed Mostefaoui, editors, *Proceedings of the 15th ACM International Symposium on QoS and Security for Wireless and Mobile Networks, Q2SWinet 2019, Miami Beach, FL, USA, November 25-29, 2019*, pages 85–91. ACM, 2019.

[16] Noura Aljeri and Azzedine Boukerche. Movement prediction models for vehicular networks: an empirical analysis. *Wirel. Networks*, 25(4):1505–1518, 2019.

[17] Noura Aljeri and Azzedine Boukerche. A novel online machine learning based RSU prediction scheme for intelligent vehicular networks. In *16th IEEE/ACS International Conference on Computer Systems and Applications, AICCSA 2019, Abu Dhabi, UAE, November 3-7, 2019*, pages 1–8. IEEE Computer Society, 2019.

[18] Noura Aljeri and Azzedine Boukerche. An optimized link duration-based mobility management scheme for connected vehicular networks. In Mónica Aguilar-Igartua, Luis J. de la Cruz Llopis, and Ahmad Mohamad Mezher, editors, *Proceedings of the 16th ACM International Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks, PE-WASUN 2019, Miami Beach, FL, USA, November 25-29, 2019*, pages 7–14. ACM, 2019.

[19] Noura Aljeri and Azzedine Boukerche. A probabilistic neural network-based road side unit prediction scheme for autonomous driving. In *2019 IEEE International Conference on Communications, ICC 2019, Shanghai, China, May 20-24, 2019*, pages 1–6. IEEE, 2019.

[20] Noura Aljeri and Azzedine Boukerche. A two-tier machine learning-based handover management scheme for intelligent vehicular networks. *Ad Hoc Networks*, 94, 2019.

[21] Noura Aljeri and Azzedine Boukerche. An adaptive traffic-flow based controller deployment scheme for software-defined vehicular networks. In Mónica Aguilar-Igartua, Paolo Bellavista, Antonio A. F. Loureiro, and Carlo Giannelli, editors, *MSWiM '20: 23rd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, Alicante, Spain, November 16-20, 2020*, pages 191–198. ACM, 2020.

[22] Noura Aljeri and Azzedine Boukerche. ADVICE-LOC: an adaptive vehicle-centric location management scheme for intelligent connected cars. *Ad Hoc Networks*, 107:102223, 2020.

[23] Noura Aljeri and Azzedine Boukerche. A dynamic MAP discovery and selection scheme for predictive hierarchical mipv6 in vehicular networks. *IEEE Trans. Veh. Technol.*, 69(1):793–806, 2020.

[24] Noura Aljeri and Azzedine Boukerche. Fog-enabled vehicular networks: A new challenge for mobility management. *Internet Technol. Lett.*, 3(6), 2020.

[25] Noura Aljeri and Azzedine Boukerche. Mobility management in 5g-enabled vehicular networks: Models, protocols, and classification. *ACM Comput. Surv.*, 53(5):92:1–92:35, 2020.

[26] Noura Aljeri and Azzedine Boukerche. A performance evaluation of time-series mobility prediction for connected vehicular networks. In Cheng Li and Ahmed Mostefaoui, editors, *Q2SWinet '20: Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks, Alicante, Spain, November 16-20, 2020*, pages 127–131. ACM, 2020.

[27] M. Almi'ani, A. A. Ghazleh, A. Al-Rahayfeh, and A. Razaque. Intelligent intrusion detection system using clustered self organized map. In *2018 Fifth International Conference on Software Defined Systems (SDS)*, pages 138–144, April 2018.

[28] Moayad Aloqaily, Safa Otoum, Ismaeel Al Ridhawi, and Yaser Jararweh. An intrusion detection system for connected vehicles in smart cities. *Ad Hoc Networks*, 90:101842, 2019. Recent advances on security and privacy in Intelligent Transportation Systems.

[29] Moayad Aloqaily, Safa Otoum, Ismaeel Al Ridhawi, and Yaser Jararweh. An intrusion detection system for connected vehicles in smart cities. *Ad Hoc Networks*, 90:101842, 2019. Recent advances on security and privacy in Intelligent Transportation Systems.

[30] Thanasis Antoniou, Ioannis Chatzigiannakis, Georgios Mylonas, Sotiris E. Nikoletseas, and Azzedine Boukerche. A new energy efficient and fault-tolerant protocol for data propagation in smart dust networks using varying transmission range. In *Proceedings 37th Annual Simulation Symposium (ANSS-37 2004), 18-22 April 2004, Arlington, VA, USA*, pages 43–52. IEEE Computer Society, 2004.

[31] David Arthur and Sergei Vassilvitskii. How slow is the k-means method? In *Proceedings of the Twenty-Second Annual Symposium on Computational Geometry*, SCG '06, page 144–153, New York, NY, USA, 2006. Association for Computing Machinery.

[32] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 284–293. PMLR, 2018.

[33] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 284–293. PMLR, 2018.

[34] B. G. Atli, Y. Miche, and A. Jung. Network intrusion detection using flow statistics. In *2018 IEEE Statistical Signal Processing Workshop (SSP)*, pages 70–74, June 2018.

[35] Mariette Awad and Rahul Khanna. *Support Vector Machines for Classification*, pages 39–66. Apress, Berkeley, CA, 2015.

[36] Athanasios Bamis, Azzedine Boukerche, Ioannis Chatzigiannakis, and Sotiris E. Nikoletseas. A mobility aware protocol synthesis for efficient routing in ad hoc mobile networks. *Comput. Networks*, 52(1):130–154, 2008.

[37] Remigiusz Baran, Andrzej Glowacz, and Andrzej Matiolanski. The efficient real- and non-real-time make and model recognition of cars. *Multimedia Tools and Applications*, pages 1–20, 2013.

[38] BenQ US. What is color gamut, August 2019. Last accessed June 6, 2021.

[39] Noppakun Boonsim and Simant Prakoonwit. Car make and model recognition under limited lighting conditions at night. *Pattern Analysis and Applications*, pages 1–13, 2016.

[40] L. Bottou, O. Chapelle, D. DeCoste, and J. Weston. *Support Vector Machine Solvers*, pages 1–27. 2007.

[41] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In Yves Lechevallier and Gilbert Saporta, editors, *19th International Conference on Computational Statistics, COMPSTAT 2010, Paris, France, August 22-27, 2010 - Keynote, Invited and Contributed Papers*, pages 177–186. Physica-Verlag, 2010.

[42] A. Boukerche, Sungbum Hong, and T. Jacob. An efficient synchronization scheme of multimedia streams in wireless and mobile systems. *Parallel and Distributed Systems, IEEE Transactions on*, 13(9):911–923, Sep 2002.

[43] A. Boukerche and Xu Li. An agent-based trust and reputation management scheme for wireless sensor networks. In *Global Telecommunications Conference, 2005. GLOBECOM '05. IEEE*, volume 3, pages 5 pp.–, Nov 2005.

[44] A. Boukerche and S. Nikoletseas. Protocols for data propagation in wireless sensor networks. In M. Guizani, editor, *Wireless Communications Systems and Networks. Information Technology: Transmission, Processing and Storage.*, pages 23–51. Springer, Boston, 2004.

[45] A. Boukerche, H.A.B. Oliveira, E.F. Nakamura, and A.A.F. Loureiro. Localization systems for wireless sensor networks. *Wireless Communications, IEEE*, 14(6):6–12, December 2007.

[46] A. Boukerche and Yonglin Ren. A secure mobile healthcare system using trust-based multicast scheme. *Selected Areas in Communications, IEEE Journal on*, 27(4):387–399, May 2009.

[47] A. Boukerche and D. Turgut. Secure time synchronization protocols for wireless sensor networks. *Wireless Communications, IEEE*, 14(5):64–69, October 2007.

[48] Azzedine Boukerche. A simulation based study of on-demand routing protocols for ad hoc wireless networks. In *Proceedings 34th Annual Simulation Symposium (SS 2001), Seattle, WA, USA, 22-26 April 2001*, pages 85–92. IEEE Computer Society, 2001.

[49] Azzedine Boukerche, editor. *Handbook of Algorithms for Wireless Networking and Mobile Computing*. Chapman and Hall/CRC, 2005.

[50] Azzedine Boukerche, editor. *Algorithms and protocols for wireless sensor networks*. John Wiley Sons, 2008.

[51] Azzedine Boukerche and Kaouther Abrougui. An efficient leader election protocol for wireless quasi-static mesh networks: Proof of correctness. In *Proceedings of IEEE International Conference on Communications, ICC 2007, Glasgow, Scotland, UK, 24-28 June 2007*, pages 3491–3496. IEEE, 2007.

[52] Azzedine Boukerche, Noura Aljeri, Kaouther Abrougui, and Yan Wang. Towards a secure hybrid adaptive gateway discovery mechanism for intelligent transportation systems. *Secur. Commun. Networks*, 9(17):4027–4047, 2016.

[53] Azzedine Boukerche, Xiuzhen Cheng, and Joseph Linus. A performance evaluation of a novel energy-aware data-centric routing algorithm in wireless sensor networks. *Wirel. Networks*, 11(5):619–635, 2005.

[54] Azzedine Boukerche, Jan M. Correa, Alba Cristina Magalhaes Melo, and Ricardo P. Jacobi. A hardware accelerator for the fast retrieval of dialign biological sequence alignments in linear space. *IEEE Transactions on Computers*, 59(6):808–821, 2010.

[55] Azzedine Boukerche and Amir Darehshoorzadeh. Opportunistic routing in wireless networks: Models, algorithms, and classifications. *ACM Comput. Surv.*, 47(2), November 2014.

[56] Azzedine Boukerche and Sajal K. Das. Dynamic load balancing strategies for conservative parallel simulations. In Alois Ferscha, Rassul Ayani, and Carl Tropper, editors, *Proceedings of the Eleventh Workshop on Parallel and Distributed Simulation, PADS '97, Lockenhaus, Austria, June 10-13, 1997*, pages 20–28. IEEE Computer Society, 1997.

[57] Azzedine Boukerche, Sajal K. Das, and Alessandro Fabbri. Analysis of a randomized congestion control scheme with DSDV routing in ad hoc wireless networks. *J. Parallel Distributed Comput.*, 61(7):967–995, 2001.

[58] Azzedine Boukerche, Sajal K. Das, and Alessandro Fabbri. Swimnet: A scalable parallel simulation testbed for wireless and mobile networks. *Wirel. Networks*, 7(5):467–486, 2001.

[59] Azzedine Boukerche, Sajal K. Das, Alessandro Fabbri, and Oktay Yildiz. Exploiting model independence for parallel PCS network simulation. In Richard M. Fujimoto and Stephen John Turner, editors, *Proceedings of the Thirteenth Workshop on Parallel and Distributed Simulation, PADS '99, Atlanta, GA, USA, May 1-4, 1999*, pages 166–173. IEEE Computer Society, 1999.

[60] Azzedine Boukerche, Horacio A. B. F. de Oliveira, Eduardo Freire Nakamura, and Antonio A. F. Loureiro. Secure localization algorithms for wireless sensor networks. *IEEE Commun. Mag.*, 46(4):96–101, 2008.

[61] Azzedine Boukerche, Yan Du, Jing Feng, and Richard Werner Nelem Pazzi. A reliable synchronous transport protocol for wireless image sensor networks. In *Proceedings of the 13th IEEE Symposium on Computers and Communications (ISCC 2008), July 6-9, Marrakech, Morocco*, pages 1083–1089. IEEE Computer Society, 2008.

[62] Azzedine Boukerche and Caron Dzermajko. Performance evaluation of data distribution management strategies. *Concurr. Pract. Exp.*, 16(15):1545–1573, 2004.

[63] Azzedine Boukerche, Khalil El-Khatib, Li Xu, and Larry Korba. SDAR: A secure distributed anonymous routing protocol for wireless and mobile ad hoc networks. In *29th Annual IEEE Conference on Local Computer Networks (LCN 2004), 16-18 November 2004, Tampa, FL, USA, Proceedings*, pages 618–624. IEEE Computer Society, 2004.

[64] Azzedine Boukerche, Khalil El-Khatib, Li Xu, and Larry Korba. An efficient secure distributed anonymous routing protocol for mobile and wireless ad hoc networks. *Comput. Commun.*, 28(10):1193–1203, 2005.

[65] Azzedine Boukerche and Xin Fei. A voronoi approach for coverage protocols in wireless sensor networks. In *Proceedings of the Global Communications Conference, 2007. GLOBECOM '07, Washington, DC, USA, 26-30 November 2007*, pages 5190–5194. IEEE, 2007.

[66] Azzedine Boukerche, Xin Fei, and Regina Borges de Araujo. An optimal coverage-preserving scheme for wireless sensor networks based on local information exchange. *Comput. Commun.*, 30(14-15):2708–2720, 2007.

[67] Azzedine Boukerche, Sungbum Hong, and Tom Jacob. A distributed algorithm for dynamic channel allocation. *Mob. Networks Appl.*, 7(2):115–126, 2002.

[68] Azzedine Boukerche and Zhijun Hou. Object detection using deep learning methods in traffic scenarios. *ACM Computing Surveys*, 54(2), March 2021.

[69] Azzedine Boukerche, Kathia Regina L. Jucá, João Bosco M. Sobral, and Mirela Sechi Moretti Annoni Notare. An artificial immune based intrusion detection model for computer and telecommunication systems. *Parallel Comput.*, 30(5-6):629–646, 2004.

[70] Azzedine Boukerche, Kathia Regina L. Jucá, João Bosco M. Sobral, and Mirela Sechi Moretti Annoni Notare. An artificial immune based intrusion detection model for computer and telecommunication systems. *Parallel Comput.*, 30(5-6):629–646, 2004.

[71] Azzedine Boukerche, Kathia Regina Lemos Jucá, Mirela Sechi Moretti Annoni Notare, and João Bosco Mangueira Sobral. Biological inspired based intrusion detection models for mobile telecommunication systems. In Stephan Olariu and Albert Y. Zomaya, editors, *Handbook of Bioinspired Algorithms and Applications*. Chapman and Hall/CRC, 2005.

[72] Azzedine Boukerche, Kathia Regina Lemos Jucá, João Bosco Sobral, and Mirela Sechi Moretti Annoni Notare. An artificial immune based intrusion detection model for computer and telecommunication systems. *Parallel Comput.*, 30(5–6):629–646, May 2004.

[73] Azzedine Boukerche, Renato B. Machado, Kathia R. L. Jucá, João Bosco M. Sobral, and Mirela S. M. A. Notare. An agent based and biological inspired real-time intrusion detection and security model for computer network operations. *Comput. Commun.*, 30(13):2649–2660, September 2007.

[74] Azzedine Boukerche, Alexander Magnano, and Noura Aljeri. Mobile IP handover for vehicular networks: Methods, models, and classifications. *ACM Comput. Surv.*, 49(4):73:1–73:34, 2017.

[75] Azzedine Boukerche, Anahit Martirosyan, and Richard Pazzi. An inter-cluster communication based energy aware and fault tolerant protocol for wireless sensor networks. *Mob. Netw. Appl.*, 13(6):614–626, December 2008.

[76] Azzedine Boukerche, Nathan J. McGraw, Caron Dzermajko, and Kaiyuan Lu. Grid-filtered region-based data distribution management in large-scale distributed simulation systems. In *Proceedings 38th Annual Simulation Symposium (ANSS-38 2005), 4-6 April 2005, San Diego, CA, USA*, pages 259–266. IEEE Computer Society, 2005.

[77] Azzedine Boukerche and Mirela Sechi Moretti Annoni Notare. Behavior-based intrusion detection in mobile phone systems. *J. Parallel Distributed Comput.*, 62(9):1476–1490, 2002.

[78] Azzedine Boukerche and Mirela Sechi Moretti Annoni Notare. Behavior-based intrusion detection in mobile phone systems. *J. Parallel Distributed Comput.*, 62(9):1476–1490, 2002.

[79] Azzedine Boukerche, Horácio A. B. F. Oliveira, Eduardo Freire Nakamura, and Antonio Alfredo Ferreira Loureiro. Dv-loc: a scalable localization protocol using voronoi diagrams for wireless sensor networks. *IEEE Wirel. Commun.*, 16(2):50–55, 2009.

[80] Azzedine Boukerche, Horacio A.B.F. Oliveira, Eduardo F. Nakamura, and Antonio A.F. Loureiro. Vehicular Ad Hoc Networks: A New Challenge for Localization-Based Systems . *Computer Communications* , 31(12):2838 – 2849, 2008. Mobility Protocols for ITS/VANET.

[81] Azzedine Boukerche, Richard Werner Nelem Pazzi, and Regina Borges Araujo. A fast and reliable protocol for wireless sensor networks in critical conditions monitoring applications. In *Proceedings of the 7th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWiM '04, pages 157–164, New York, NY, USA, 2004. ACM.

[82] Azzedine Boukerche, Richard Werner Nelem Pazzi, and Regina Borges de Araujo. HPEQ A hierarchical periodic, event-driven and query-based wireless sensor network protocol. In *30th Annual IEEE Conference on Local Computer Networks (LCN 2005), 15-17 November 2005, Sydney, Australia, Proceedings*, pages 560–567. IEEE Computer Society, 2005.

[83] Azzedine Boukerche, Richard Werner Nelem Pazzi, and Jing Feng. An end-to-end virtual environment streaming technique for thin mobile devices over heterogeneous networks. *Comput. Commun.*, 31(11):2716–2725, 2008.

[84] Azzedine Boukerche and Yonglin Ren. A security management scheme using a novel computational reputation model for wireless and mobile ad hoc networks. In Luciano Bononi and Isabelle Guérin Lassous, editors, *Proceedings of the 5th ACM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks, PE-WASUN 2008, Vancouver, British Columbia, Canada, October 27-28, 2008*, pages 88–95. ACM, 2008.

[85] Azzedine Boukerche and Yonglin Ren. A trust-based security system for ubiquitous and pervasive computing environments. *Computer Communications*, 31(18):4343 –

4351, 2008. Secure Multi-Mode Systems and their Applications for Pervasive Computing.

[86] Azzedine Boukerche, Cristiano G. Rezende, and Richard Werner Nelem Pazzi. Improving neighbor localization in vehicular ad hoc networks to avoid overhead from periodic messages. In *Proceedings of the Global Communications Conference, 2009. GLOBECOM 2009, Honolulu, Hawaii, USA, 30 November - 4 December 2009*, pages 1–6. IEEE, 2009.

[87] Azzedine Boukerche and Steve Rogers. Performance of GZRP ad hoc routing protocol. *J. Interconnect. Networks*, 2(1):31–48, 2001.

[88] Azzedine Boukerche and Amber Roy. Dynamic grid-based approach to data distribution management. *J. Parallel Distributed Comput.*, 62(3):366–392, 2002.

[89] Azzedine Boukerche, Amber Roy, and Neville Thomas. Dynamic grid-based multicast group assignment in data distribution management. In *4th International Workshop on Distributed Simulation and Real-Time Applications (DS-RT 2000), 25-17 August 2000, San Francisco, CA, USA*, pages 47–54. IEEE Computer Society, 2000.

[90] Azzedine Boukerche and Samer Samarah. An efficient data extraction mechanism for mining association rules from wireless sensor networks. In *Proceedings of IEEE International Conference on Communications, ICC 2007, Glasgow, Scotland, UK, 24-28 June 2007*, pages 3936–3941. IEEE, 2007.

[91] Azzedine Boukerche and Samer Samarah. A novel algorithm for mining association rules in wireless ad hoc sensor networks. *IEEE Trans. Parallel Distributed Syst.*, 19(7):865–877, 2008.

[92] Azzedine Boukerche, Abdul Jabbar Siddiqui, and Abdelhamid Mammeri. Automated vehicle detection and classification: Models, methods, and techniques. *ACM Comput. Surv.*, 50(5), October 2017.

[93] Azzedine Boukerche and Carl Tropper. A static partitioning and mapping algorithm for conservative parallel simulations. In *Proceedings of the Eighth Workshop on Par-*

allel and Distributed Simulation, PADS '94, page 164–172, New York, NY, USA, 1994. Association for Computing Machinery.

[94] Azzedine Boukerche, Anis Zarrad, and Regina Borges de Araujo. A cross-layer approach-based gnutella for collaborative virtual environments over mobile ad hoc networks. *IEEE Trans. Parallel Distributed Syst.*, 21(7):911–924, 2010.

[95] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, Aug 1996.

[96] Tom B. Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch, 2017.

[97] Christopher J. C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, June 1998.

[98] J. Canedo and A. Skjellum. Using machine learning to secure iot systems. In *2016 14th Annual Conference on Privacy, Security and Trust (PST)*, pages 219–222, Dec 2016.

[99] Rich Caruana. Multitask learning. *Mach. Learn.*, 28(1):41–75, July 1997.

[100] Clayson Celes, Fabrício A. Silva, Azzedine Boukerche, Rossana Maria de Castro Andrade, and Antonio A. F. Loureiro. Improving VANET simulation with calibrated vehicular mobility traces. *IEEE Trans. Mob. Comput.*, 16(12):3376–3389, 2017.

[101] G. Chen, J. Yang, H. Jin, J. Brandt, E. Shechtman, A. Agarwala, and T. X. Han. Large-scale visual font recognition. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3598–3605, June 2014.

[102] Li-Chih Chen, Jun-Wei Hsieh, Yilin Yan, and Duan-Yu Chen. Vehicle Make and Model Recognition using Sparse Representation and Symmetrical SURFs. *Pattern Recognition*, 48(6):1979 – 1998, 2015.

[103] Qian Chen, Gautam Srivastava, Reza M. Parizi, Moayad Aloqaily, and Ismaeel Al Ridhawi. An incentive-aware blockchain-based solution for internet of fake media things. *Information Processing Management*, 57(6):102370, 2020.

[104] Xavier Clady, Pablo Negri, Maurice Milgram, and Raphael Poulenard. Multi-class vehicle type recognition system. In Lionel Prevost, Simone Marinai, and Friedhelm Schwenker, editors, *Artificial Neural Networks in Pattern Recognition*, volume 5064 of *Lecture Notes in Computer Science*, pages 228–239. Springer Berlin Heidelberg, 2008.

[105] Sergio Correia, Azzedine Boukerche, and Rodolfo Ipolito Meneguette. An architecture for hierarchical software-defined vehicular networks. *IEEE Commun. Mag.*, 55(7):80–86, 2017.

[106] Andrei Costin. Security of cctv and video surveillance systems: Threats, vulnerabilities, attacks, and mitigations. TrustED '16, page 45–54, New York, NY, USA, 2016. Association for Computing Machinery.

[107] Rodolfo W. L. Coutinho, Azzedine Boukerche, Luiz Filipe M. Vieira, and Antonio A. F. Loureiro. GEDAR: geographic and opportunistic routing protocol with depth adjustment for mobile underwater sensor networks. In *IEEE International Conference on Communications, ICC 2014, Sydney, Australia, June 10-14, 2014*, pages 251–256. IEEE, 2014.

[108] Rodolfo W. L. Coutinho, Azzedine Boukerche, Luiz Filipe M. Vieira, and Antonio A. F. Loureiro. Design guidelines for opportunistic routing in underwater networks. *IEEE Commun. Mag.*, 54(2):40–48, 2016.

[109] Rodolfo W. L. Coutinho, Azzedine Boukerche, Luiz Filipe M. Vieira, and Antonio Alfredo Ferreira Loureiro. A novel void node recovery paradigm for long-term underwater sensor networks. *Ad Hoc Networks*, 34:144–156, 2015.

[110] Rodolfo Wanderson Lima Coutinho, Azzedine Boukerche, Luiz Filipe Menezes Vieira, and Antonio Alfredo Ferreira Loureiro. Geographic and opportunistic routing for underwater sensor networks. *IEEE Trans. Computers*, 65(2):548–561, 2016.

[111] Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual Categorization with Bags of Keypoints. In *Workshop on Statistical Learning in Computer Vision, ECCV*, pages 1–22, 2004.

[112] Amir Darehshoorzadeh and Azzedine Boukerche. Underwater sensor networks: a new challenge for opportunistic routing protocols. *IEEE Commun. Mag.*, 53(11):98–107, 2015.

[113] Nilaksh Das, Madhuri Shanbhogue, Shang-Tse Chen, Fred Hohman, Siwei Li, Li Chen, Michael E. Kounavis, and Duen Horng Chau. Shield: Fast, practical defense and vaccination for deep learning using jpeg compression. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, page 196–204, New York, NY, USA, 2018. Association for Computing Machinery.

[114] H.A.B.F. de Oliveira, A. Boukerche, E. Freire Nakamura, and A.A.F. Loureiro. An efficient directed localization recursion protocol for wireless sensor networks. *IEEE Transactions on Computers*, 58(5):677–691, May 2009.

[115] Horacio A. B. F. de Oliveira, Eduardo Freire Nakamura, Antonio Alfredo Ferreira Loureiro, and Azzedine Boukerche. Error analysis of localization systems for sensor networks. In Cyrus Shahabi and Omar Boucelma, editors, *GIS*, pages 71–78. ACM, 2005.

[116] Ranjie Duan, Xingjun Ma, Yisen Wang, James Bailey, A. Kai Qin, and Yun Yang. Adversarial camouflage: Hiding physical-world attacks with natural styles. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020, pages 997–1005. IEEE, 2020.

[117] Mourad Elhadef, Azzedine Boukerche, and Hisham Elkadiki. Diagnosing mobile ad-hoc networks: two distributed comparison-based self-diagnosis protocols. In Albert Y. Zomaya and Luciano Bononi, editors, *Proceedings of the Forth ACM International Workshop on Mobility Management & Wireless Access, MOBIWAC 2006, Terromolinos, Spain, October 2, 2006*, pages 18–27. ACM, 2006.

[118] Mourad Elhadef, Azzedine Boukerche, and Hisham Elkadiki. Performance analysis of a distributed comparison-based self-diagnosis protocol for wireless ad-hoc networks. In Enrique Alba, Carla-Fabiana Chiasserini, Nael B. Abu-Ghazaleh, and Renato Lo Cigno, editors, *Proceedings of the 9th International Symposium on Modeling Analysis*

and Simulation of Wireless and Mobile Systems, MSWiM 2006, Terromolinos, Spain, October 2-6, 2006, pages 165–172. ACM, 2006.

[119] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 1625–1634. IEEE Computer Society, 2018.

[120] F. Farahnakian and J. Heikkonen. A deep auto-encoder based approach for intrusion detection system. In *2018 20th International Conference on Advanced Communication Technology (ICACT)*, pages 178–183, Feb 2018.

[121] A. Farhadi, D. Hoiem, D. Forsyth, and I. Endres. Describing objects by their attributes. *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshops)*, 00:1778–1785, 2009.

[122] P. Foggia, N. Petkov, A. Saggese, N. Strisciuglio, and M. Vento. Audio surveillance of roads: A system for detecting anomalous sounds. *IEEE Transactions on Intelligent Transportation Systems*, 17(1):279–288, 2016.

[123] P. Foggia, A. Saggese, N. Strisciuglio, M. Vento, and N. Petkov. Car crashes detection by audio analysis in crowded roads. In *2015 12th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6, 2015.

[124] Baraq Ghaleb, Ahmed Yassin Al-Dubai, Elias Ekonomou, Ayoub Alsarhan, Youssef Nasser, Lewis M. Mackenzie, and Azzedine Boukerche. A survey of limitations and enhancements of the ipv6 routing protocol for low-power and lossy networks: A focus on core operations. *IEEE Commun. Surv. Tutorials*, 21(2):1607–1635, 2019.

[125] Gerard Drapper Gil, Arash Habibi Lashkari, Mohammad Mamun, and Ali A. Ghorbani. Characterization of encrypted and vpn traffic using time-related features. In *2nd International Conference on Information Systems Security and Privacy (ICISSP 2016)*, pages 407–414, 2016.

[126] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[127] Panagiotis I. Radoglou Grammatikis, Panagiotis G. Sarigiannidis, and Ioannis D. Moscholios. Securing the internet of things: Challenges, threats and solutions. *Internet of Things*, 5:41 – 70, 2019.

[128] Chuan Guo, Mayank Rana, M. Cissé, and L. V. D. Maaten. Countering adversarial images using input transformations. *ArXiv*, abs/1711.00117, 2018.

[129] Chuan Guo, Mayank Rana, Moustapha Cissé, and Laurens van der Maaten. Countering adversarial images using input transformations. In *6th International Conference on Learning Representations, ICLR (Poster)*, 2018.

[130] M. A. Habib, M. Ahmad, S. Jabbar, S. H. Ahmed, and J. J. P. C. Rodrigues. Speeding up the internet of things: Leaiot: A lightweight encryption algorithm toward low-latency communication for the internet of things. *IEEE Consumer Electronics Magazine*, 7(6):31–37, Nov 2018.

[131] Hadi Habibzadeh, Tolga Soyata, Burak Kantarci, Azzedine Boukerche, and Cem Kaptan. Sensing, communication and security planes: A new challenge for a smart city system design. *Comput. Networks*, 144:163–200, 2018.

[132] J. Hayes. On visible adversarial perturbations digital watermarking. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1678–16787, 2018.

[133] D. He, S. Chan, X. Ni, and M. Guizani. Software-defined-networking-enabled traffic anomaly detection and mitigation. *IEEE Internet of Things Journal*, 4(6):1890–1898, Dec 2017.

[134] H. He, Z. Shao, and J. Tan. Recognition of Car Makes and Models From a Single Traffic-Camera Image. *IEEE Transactions on Intelligent Transportation Systems*, PP(99):1–11, 2015.

[135] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016.

[136] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. *CoRR*, abs/1603.05027, 2016.

[137] Geoffrey E. Hinton and Richard S. Zemel. Autoencoders, minimum description length and helmholtz free energy. In *Proceedings of the 6th International Conference on Neural Information Processing Systems*, NIPS'93, pages 3–10, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[138] Jianwei Hou, Leilei Qu, and Wenchang Shi. A survey on internet of things security from data perspectives. *Computer Networks*, 148:295 – 306, 2019.

[139] Jun-Wei Hsieh, Li-Chih Chen, and Duan-Yu Chen. Symmetrical SURF and Its Applications to Vehicle Detection and Vehicle Make and Model Recognition. *IEEE Transactions on Intelligent Transportation Systems*, 15(1):6–20, Feb 2014.

[140] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. Deep networks with stochastic depth. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 646–661, Cham, 2016. Springer International Publishing.

[141] Lifeng Huang, Chengying Gao, Yuyin Zhou, Cihang Xie, Alan L. Yuille, Changqing Zou, and Ning Liu. Universal physical camouflage attacks on object detectors. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020, pages 717–726. IEEE, 2020.

[142] Ling Huang, Anthony D. Joseph, Blaine Nelson, Benjamin I.P. Rubinstein, and J. D. Tygar. Adversarial machine learning. In *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*, AISec '11, page 43–58, New York, NY, USA, 2011. Association for Computing Machinery.

[143] Christiana Ioannou and Vasos Vassiliou. An intrusion detection system for constrained wsn and iot nodes based on binary logistic regression. In *Proceedings of the 21st ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWIM '18, page 259–263, New York, NY, USA, 2018. Association for Computing Machinery.

[144] D.M. Jang and M. Turk. Car-rec: A real time car recognition system. In *IEEE Workshop on Applications of Computer Vision (WACV)*, pages 599–605, Jan 2011.

[145] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross B. Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *CoRR*, abs/1408.5093, 2014.

[146] Naor Kalbo, Yisroel Mirsky, Asaf Shabtai, and Yuval Elovici. The security of ip-based video surveillance systems. *Sensors*, 20(17):4806, Aug 2020.

[147] Kaspersky. Kaspersky lab ddos intelligence quarterly report: amplification attacks and old botnets make a comeback, April 2018. [Last accessed 29-October-2018].

[148] Rana Abou Khamis, M. Omair Shafiq, and Ashraf Matrawy. Investigating resistance of deep learning-based IDS against adversaries using min-max optimization. In *2020 IEEE International Conference on Communications, ICC 2020, Dublin, Ireland, June 7-11, 2020*, pages 1–7. IEEE, 2020.

[149] Yongsu Kim, Hyoeun Kang, Afifatul Mukaroh, Naufal Suryanto, Harashta Tatimma Larasati, and Howon Kim. Spatially localized perturbation gan (slp-gan) for generating invisible adversarial patches. In Ilsun You, editor, *Information Security Applications*, pages 3–15, Cham, 2020. Springer International Publishing.

[150] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[151] Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou, and Jeffrey Voas. Ddos in the iot: Mirai and other botnets. *Computer*, 50(7):80–84, 2017.

[152] J. Krause, H. Jin, J. Yang, and L. Fei-Fei. Fine-grained recognition without part annotations. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5546–5555, June 2015.

[153] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3d object representations for fine-grained categorization. In *2013 IEEE International Conference on Computer Vision Workshops*, pages 554–561, Dec 2013.

[154] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pages 1106–1114, 2012.

[155] A. Ranjith Kumar and A. Sivagami. Security aware multipath routing protocol for wmsns for minimizing effect of compromising attacks. *J. Netw. Syst. Manag.*, 27(3):573–599, 2019.

[156] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net, 2017.

[157] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[158] Youssef Lahrouni, Caroly Pereira, Boucif Amar Bensaber, and Ismaïl Biskri. Using mathematical methods against denial of service (dos) attacks in VANET. In *15th ACM International Symposium on Mobility Management and Wireless Access, MOBIWAC 2017*, pages 17–22. ACM, 2017.

[159] C. H. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 951–958, June 2009.

[160] HyoJong Lee. Neural Network Approach to Identify Model of Vehicles. In Jun Wang, Zhang Yi, JacekM. Zurada, Bao-Liang Lu, and Hujun Yin, editors, *Advances in Neural Networks - ISNN 2006*, volume 3973 of *Lecture Notes in Computer Science*, pages 66–72. Springer Berlin Heidelberg, 2006.

[161] W. Lee, A. Rezapour, and W. Tzeng. Monsieur poirot: Detecting botnets using re-identification algorithm and nontrivial feature selection technique. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–6, May 2018.

[162] J. Li, Z. Zhao, R. Li, H. Zhang, and T. Zhang. Ai-based two-stage intrusion detection for software defined iot networks. *IEEE Internet of Things Journal*, pages 1–1, 2018.

[163] Jing Li, Weifa Liang, Wenzheng Xu, Zichuan Xu, and Jin Zhao. Maximizing the quality of user experience of using services in edge computing for delay-sensitive iot applications. In *23rd Int'l ACM Conf. on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 113–121. ACM, 2020.

[164] Juncheng Li, Frank R. Schmidt, and J. Zico Kolter. Adversarial camera stickers: A physical camera-based attack on deep learning systems. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 3896–3904. PMLR, 2019.

[165] Wei-Chao Lin, Shih-Wen Ke, and Chih-Fong Tsai. Cann: An intrusion detection system based on combining cluster centers and nearest neighbors. *Knowledge-Based Systems*, 78:13 – 21, 2015.

[166] Xin Liu, Huanrui Yang, Ziwei Liu, Linghao Song, Yiran Chen, and Hai Li. DPATCH: an adversarial patch attack on object detectors. In Huáscar Espinoza,

Seán Ó hÉigeartaigh, Xiaowei Huang, José Hernández-Orallo, and Mauricio Castillo-Effen, editors, *Workshop on Artificial Intelligence Safety 2019 co-located with the Thirty-Third AAAI Conference on Artificial Intelligence 2019 (AAAI-19), Honolulu, Hawaii, January 27, 2019*, volume 2301 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2019.

[167] D.F. Llorca, D. Colás, I.G. Daza, I. Parra, and M.A. Sotelo. Vehicle model recognition using geometry and appearance of car emblems from rear view images. In *17th IEEE International Conference on Intelligent Transportation Systems*, pages 3094–3099, Oct 2014.

[168] M. Fraz and E. A. Edirisinghe and M. S. Sarfraz . Mid-level-Representation Based Lexicon for Vehicle Make and Model Recognition. In *22nd International Conference on Pattern Recognition (ICPR)*, pages 393–398, Aug 2014.

[169] Renato B. Machado, Azzedine Boukerche, João Bosco M. Sobral, Kathia Regina L. Jucá, and Mirela Sechi Moretti Annoni Notare. A hybrid artificial immune and mobile agent intrusion detection based model for computer network operations. In *19th International Parallel and Distributed Processing Symposium (IPDPS 2005), CD-ROM / Abstracts Proceedings, 4-8 April 2005, Denver, CO, USA*. IEEE Computer Society, 2005.

[170] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5188–5196, 2015.

[171] Abdelhamid Mammeri, Azzedine Boukerche, and Zongzhi Tang. A real-time lane marking localization, tracking and communication system. *Comput. Commun.*, 73:132–143, 2016.

[172] N. Marir, H. Wang, G. Feng, B. Li, and M. Jia. Distributed abnormal behavior detection approach based on deep belief network and ensemble svm using spark. *IEEE Access*, 6:59657–59671, 2018.

[173] Tony Marteau, Sitou Afanou, David Sodoyer, Sébastien Ambellouis, and Fouzia Boukour. Audio events detection in noisy embedded railway environments. In Simona Bernardi, Valeria Vittorini, Francesco Flammini, Roberto Nardone, Stefano Marrone, Rasmus Adler, Daniel Schneider, Philipp Schleiß, Nicola Nostro, Rasmus Løvenstein Olsen, Amleto Di Salle, and Paolo Masci, editors, *Dependable Computing - EDCC 2020 Workshops*, pages 20–32, Cham, 2020. Springer International Publishing.

[174] Anahit Martirosyan, Azzedine Boukerche, and Richard Werner Nelem Pazzi. A taxonomy of cluster-based routing protocols for wireless sensor networks. In *9th International Symposium on Parallel Architectures, Algorithms, and Networks, ISPAN 2008, 7-9 May 2008, Sydney, NSW, Australia*, pages 247–253. IEEE Computer Society, 2008.

[175] 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*, 17(3):12–22, Jul.-Sep. 2018.

[176] Mohammed Amine Merzoug, Ahmed Mostefaoui, and Abderrezak Benyahia. Smart iot notification system for efficient in-city parking. In *15th ACM Int'l Symposium on QoS and Security for Wireless and Mobile Networks, Q2SWinet 2019, Miami Beach, FL, USA, November 25-29, 2019*, pages 37–42. ACM, 2019.

[177] Mohammed Amine Merzoug, Ahmed Mostefaoui, Mohammed H. Kechout, and Sebti Tamraoui. Deep learning for resource-limited devices. In Cheng Li and Ahmed Mostefaoui, editors, *16th ACM Symposium on QoS and Security for Wireless and Mobile Networks, Alicante, Spain, November 16-20, 2020*, pages 81–87. ACM, 2020.

[178] Trend Micro. Ddos - security news - trend micro usa.

[179] M. Miettinen and A. Sadeghi. Keynote: Internet of things or threats? on building trust in iot. In *2018 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, pages 1–9, Sep. 2018.

[180] Yisroel Mirsky, Tomer Doitshman, Yuval Elovici, and Asaf Shabtai. Kitsune: An ensemble of autoencoders for online network intrusion detection. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*, 2018.

[181] A. Mohaisen and J. Kim. Securing the internet of things: A machine learning approach, May 2018. [2018 IEEE International Conference on Communications Tutorials ; Last accessed 29-October-2018].

[182] Christoph Molnar. *Interpretable Machine Learning - A Guide for Making Black Box Models Explainable*. 2020.

[183] Raha Moraffah, Mansooreh Karami, Ruocheng Guo, Adrienne Raglin, and Huan Liu. Causal interpretability for machine learning - problems, methods and evaluation. *SIGKDD Explor. Newsl.*, 22(1):18–33, May 2020.

[184] Ahmed Mostefaoui, Mahmoud Melkemi, and Azzedine Boukerche. Localized routing approach to bypass holes in wireless sensor networks. *IEEE Trans. Computers*, 63(12):3053–3065, 2014.

[185] N. Moustafa, B. Turnbull, and K. R. Choo. An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things. *IEEE Internet of Things Journal*, pages 1–1, 2018.

[186] D. T. Munroe and M. G. Madden. Multi-Class and Single-Class Classification Approaches to Vehicle Model Recognition from Images. In *16th Irish Conference on Artificial Intelligence and Cognitive Science*, pages 93–104, Sep 2005.

[187] S. Nõmm and H. Bahṣi. Unsupervised anomaly based botnet detection in iot networks. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1048–1053, Dec 2018.

[188] Deeraj Nagothu, Yu Chen, Erik Blasch, Alexander Aved, and Sencun Zhu. Detecting malicious false frame injection attacks on surveillance systems at the edge using electrical network frequency signals. *Sensors*, 19(11), 2019.

[189] Loris Nanni and Alessandra Lumini. Heterogeneous Bag-of-Features for Object/Scene Recognition. *Applied Soft Computing*, 13(4):2171 – 2178, 2013.

[190] M. Naseer, S. Khan, and F. Porikli. Local gradients smoothing: Defense against localized adversarial attacks. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1300–1307, Los Alamitos, CA, USA, jan 2019. IEEE Computer Society.

[191] A. Nisioti, A. Mylonas, P. D. Yoo, and V. Katos. From intrusion detection to attacker attribution: A comprehensive survey of unsupervised methods. *IEEE Communications Surveys Tutorials*, pages 1–1, 2018.

[192] Ramzi A. Nofal, Nam Tran, Carlos Garcia, Yuhong Liu, and Behnam Dezfouli. A comprehensive empirical analysis of tls handshake and record layer on iot platforms. In *Proceedings of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWIM '19, page 61–70, New York, NY, USA, 2019. Association for Computing Machinery.

[193] Horacio A.B.F. Oliveira, Azzedine Boukerche, Eduardo F. Nakamura, and Antonio A.F. Loureiro. Localization in time and space for wireless sensor networks: An efficient and lightweight algorithm. *Performance Evaluation*, 66(3–5):209 – 222, 2009. Modeling and Analysis of Wireless Networks: Selected Papers from {MSWiM} 2007.

[194] Renê Oliveira, Carlos Montez, Azzedine Boukerche, and Michelle S. Wangham. Reliable data dissemination protocol for VANET traffic safety applications. *Ad Hoc Networks*, 63:30–44, 2017.

[195] Charlie Osborne and Zero Day. The most interesting internet-connected vehicle hacks on record.

[196] S. Otoum, B. Kantarci, and H. Mouftah. Empowering reinforcement learning on big sensed data for intrusion detection. In *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, pages 1–7, 2019.

[197] S. Otoum, B. Kantarci, and H. T. Mouftah. On the feasibility of deep learning in sensor network intrusion detection. *IEEE Networking Letters*, 1(2):68–71, 2019.

[198] S. Otoum, B. Kantarci, and H. T. Mouftah. A novel ensemble method for advanced intrusion detection in wireless sensor networks. In *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, pages 1–6, 2020.

[199] J. Pan and J. McElhannon. Future edge cloud and edge computing for internet of things applications. *IEEE Internet of Things Journal*, 5(1):439–449, Feb 2018.

[200] Nicolas Papernot, Patrick D. McDaniel, Ian J. Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In Ramesh Karri, Ozgur Sinanoglu, Ahmad-Reza Sadeghi, and Xun Yi, editors, *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2017, Abu Dhabi, United Arab Emirates, April 2-6, 2017*, pages 506–519. ACM, 2017.

[201] Richard W. Pazzi and Azzedine Boukerche. Propane: A progressive panorama streaming protocol to support interactive 3d virtual environment exploration on graphics-constrained devices. *ACM Trans. Multimedia Comput. Commun. Appl.*, 11(1), September 2014.

[202] Richard W.N. Pazzi and Azzedine Boukerche. Mobile data collector strategy for delay-sensitive applications over wireless sensor networks. *Computer Communications*, 31(5):1028 – 1039, 2008. Mobility Management and Wireless Access.

[203] G. Pearce and N. Pears. Automatic make and model recognition from frontal images of cars. In *8th IEEE International Conference on Advanced Video and Signal-Based Surveillance (AVSS)*, pages 373–378, Aug 2011.

[204] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[205] J. Petit, B. Stottelaar, M. Feiri, and F. Kargi. Remote attacks on automated vehicles sensors: Experiments on camera and lidar. In *Black Hat Europe 2015*, November 2015.

[206] J. Prokaj and G. Medioni. 3-d model based vehicle recognition. In *Workshop on Applications of Computer Vision (WACV)*, pages 1–7, Dec 2009.

[207] A. Psyllos, C.N. Anagnostopoulos, and E. Kayafas. Vehicle model recognition from frontal view image measurements. *Computer Standards & Interfaces*, 33(2):142 – 151, 2011. XVI IMEKO TC4 Symposium Ëxploring New Frontiers of Instrumentation and Methods for Electrical and Electronic Measurementsänd XIII International Workshop on ADC Modelling and Testing.

[208] Shahid Raza, Linus Wallgren, and Thiemo Voigt. Svelte: Real-time intrusion detection in the internet of things. *Ad Hoc Networks*, 11(8):2661 – 2674, 2013.

[209] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525, 2017.

[210] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[211] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 6517–6525. IEEE Computer Society, 2017.

[212] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[213] Huali Ren, Teng Huang, and Hongyang Yan. Adversarial examples: attacks and defenses in the physical world. *International Journal of Machine Learning and Cybernetics*, 2021.

[214] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(6):1137–1149, 2017.

[215] Yonglin Ren and A. Boukerche. Modeling and managing the trust for wireless and mobile ad hoc networks. In *Communications, 2008. ICC '08. IEEE International Conference on*, pages 2129–2133, May 2008.

[216] Yonglin Ren, R.W.N. Pazzi, and A. Boukerche. Monitoring patients via a secure and mobile healthcare system. *Wireless Communications, IEEE*, 17(1):59–65, February 2010.

[217] F. Restuccia, S. D'Oro, and T. Melodia. Securing the internet of things in the age of machine learning and software-defined networking. *IEEE Internet of Things Journal*, 5(6):4829–4842, Dec 2018.

[218] C. Rezende, A. Boukerche, H.S. Ramos, and A.A.F. Loureiro. A reactive and scalable unicast solution for video streaming over vanets. *IEEE Transactions on Computers*, 64(3):614–626, March 2015.

[219] Cristiano Rezende, Abdelhamid Mammeri, Azzedine Boukerche, and Antonio A.F. Loureiro. A receiver-based video dissemination solution for vehicular networks with content transmissions decoupled from relay node selection. *Ad Hoc Networks*, 17:1 – 17, 2014.

[220] Andras Rozsa, Ethan M. Rudd, and Terrance E. Boult. Adversarial diversity and hard positive generation. In *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2016, Las Vegas, NV, USA, June 26 - July 1, 2016*, pages 410–417. IEEE Computer Society, 2016.

[221] Daniel B. Russakoff, Carlo Tomasi, Torsten Rohlfing, and Calvin R. Maurer Jr. Image similarity using mutual information of regions. In Tomás Pajdla and Jiri Matas, editors, *Computer Vision - ECCV 2004, 8th European Conference on Computer Vision, Prague, Czech Republic, May 11-14, 2004. Proceedings, Part III*, volume 3023 of *Lecture Notes in Computer Science*, pages 596–607. Springer, 2004.

[222] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[223] Mayu Sakurada and Takehisa Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2Nd Workshop on Machine Learning for Sensory Data Analysis*, MLSDA'14, pages 4:4–4:11, New York, NY, USA, 2014. ACM.

[224] Haythem Bany Salameh, Rawan Derbas, Moayad Aloqaily, and Azzedine Boukerche. Secure routing in multi-hop iot-based cognitive radio networks under jamming attacks. In *22nd Int'l ACM Conf. on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 323–327. ACM, 2019.

[225] S. Samarah, M. Al-Hajri, and A. Boukerche. A predictive energy-efficient technique to support object-tracking sensor networks. *IEEE Transactions on Vehicular Technology*, 60(2):656–663, Feb 2011.

[226] S. Saravi and E. A. Edirisinghe. Vehicle make and model recognition in cctv footage. In *Digital Signal Processing (DSP), 2013 18th International Conference on*, pages 1–6, July 2013.

[227] H. Sedjelmaci, S. M. Senouci, and M. A. Abu-Rgheff. An efficient and lightweight intrusion detection mechanism for service-oriented vehicular networks. *IEEE Internet of Things Journal*, 1(6):570–577, Dec 2014.

[228] Kewei Sha, Wei Wei, T. Andrew Yang, Zhiwei Wang, and Weisong Shi. On security challenges and open issues in internet of things. *Future Generation Computer Systems*, 83:326 – 337, 2018.

[229] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *4th International Conference on Information Systems Security and Privacy (ICISSP)*, January 2018.

[230] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K. Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 1528–1540, New York, NY, USA, 2016. Association for Computing Machinery.

[231] S. Shasha, M. Mahmoud, M. Mannan, and A. Youssef. Playing with danger: A taxonomy and evaluation of threats to smart toys. *IEEE Internet of Things Journal*, pages 1–1, 2018.

[232] A. J. Siddiqui and A. Boukerche. Encoded flow features for network intrusion detection in internet of things. In *2020 IEEE 17th Annual Consumer Communications Networking Conference (CCNC)*, pages 1–6, 2020.

[233] Abdul Jabbar Siddiqui and Azzedine Boukerche. Adaptive ensembles of autoencoders for unsupervised iot network intrusion detection. *Computing*, 2021.

[234] Abdul Jabbar Siddiqui and Azzedine Boukerche. A novel lightweight defense method against adversarial patches-based attacks on automated vehicle make and model recognition systems. *Journal of Network and Systems Management*, 29(41), 2021.

[235] Abdul Jabbar Siddiqui, Abdelhamid Mammeri, and Azzedine Boukerche. Real-time vehicle make and model recognition based on a bag of surf features. *Trans. Intell. Transport. Sys.*, 17(11):3205–3219, November 2016.

[236] Fabrício A. Silva, Azzedine Boukerche, Thais R. M. Braga Silva, Linnyer Beatrys Ruiz, Eduardo Cerqueira, and Antonio A. F. Loureiro. Vehicular networks: A new challenge for content-delivery-based applications. *ACM Comput. Surv.*, 49(1):11:1–11:29, 2016.

[237] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[238] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[239] J. Sochor, A. Herout, and J. Havel. Boxcars: 3d boxes as cnn input for improved fine-grained vehicle recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3006–3015, June 2016.

[240] Dawn Song, Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Florian Tramèr, Atul Prakash, and Tadayoshi Kohno. Physical adversarial examples for object detectors. In Christian Rossow and Yves Younan, editors, *12th USENIX Workshop on Offensive Technologies, WOOT 2018, Baltimore, MD, USA, August 13-14, 2018*. USENIX Association, 2018.

[241] Robin Joe Prabhahar Soundar Raja James, Abdurhman Ali Albasir, Kshirasagar Naik, Marzia Zaman, and Nishith Goel. A power signal based dynamic approach to detecting anomalous behavior in wireless devices. In *Proceedings of the 16th ACM International Symposium on Mobility Management and Wireless Access*, MobiWac'18, page 9–18, New York, NY, USA, 2018. Association for Computing Machinery.

[242] Suvrit Sra, Sebastian Nowozin, and Stephen J Wright. *Optimization for Machine Learning*. MIT Press, Cambridge, MA, 2012.

[243] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[244] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Trans. Evol. Comput.*, 23(5):828–841, 2019.

[245] Peng Sun, Noura Aljeri, and Azzedine Boukerche. A novel passive road side unit detection scheme in vehicular networks. In *2017 IEEE Global Communications Conference, GLOBECOM 2017, Singapore, December 4-8, 2017*, pages 1–5. IEEE, 2017.

[246] Peng Sun, Noura AlJeri, and Azzedine Boukerche. A fast vehicular traffic flow prediction scheme based on fourier and wavelet analysis. In *IEEE Global Communications Conference, GLOBECOM 2018, Abu Dhabi, United Arab Emirates, December 9-13, 2018*, pages 1–6. IEEE, 2018.

[247] Peng Sun, Noura AlJeri, and Azzedine Boukerche. DACON: A novel traffic prediction and data-highway-assisted content delivery protocol for intelligent vehicular networks. *IEEE Trans. Sustain. Comput.*, 5(4):501–513, 2020.

[248] Peng Sun, Noura AlJeri, and Azzedine Boukerche. An energy-efficient proactive handover scheme for vehicular networks based on passive RSU detection. *IEEE Trans. Sustain. Comput.*, 5(1):37–47, 2020.

[249] Peng Sun, Noura Aljeri, and Azzedine Boukerche. Machine learning-based models for real-time traffic flow prediction in vehicular networks. *IEEE Netw.*, 34(3):178–185, 2020.

[250] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[251] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[252] Liang Tan, Huan Xiao, Keping Yu, Moayad Aloqaily, and Yaser Jararweh. A blockchain-empowered crowdsourcing system for 5g-enabled smart cities. *Computer Standards Interfaces*, 76:103517, 2021.

[253] Diya Thomas and Rajan Shankaran. A secure barrier coverage scheduling framework for wsn-based iot applications. In *23rd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 215–224. ACM, 2020.

[254] S. Thys, W. V. Ranst, and T. Goedemé. Fooling automated surveillance cameras: Adversarial patches to attack person detection. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 49–55, 2019.

[255] TrendMicro. Securing ip surveillance cameras in the iot ecosystem, April 2018. [Last accessed 06-September-2020].

[256] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[257] V. Varjas and A. Tanacs. Car recognition from frontal images in mobile environment. In *Image and Signal Processing and Analysis (ISPA), 2013 8th International Symposium on*, pages 819–823, Sept 2013.

[258] Nalam Venkata Abhishek, Anshoo Tandon, Teng Joon Lim, and Biplab Sikdar. Detecting forwarding misbehavior in clustered iot networks. In *Proceedings of the 14th ACM International Symposium on QoS and Security for Wireless and Mobile Networks*, Q2SWinet'18, page 1–6, New York, NY, USA, 2018. Association for Computing Machinery.

[259] M. Villari, M. Fazio, S. Dustdar, O. Rana, L. Chen, and R. Ranjan. Software defined membrane: Policy-driven edge and internet of things security. *IEEE Cloud Computing*, 4(4):92–99, July 2017.

[260] Renfei Wang, C. Rezende, H.S. Ramos, R.W. Pazzi, A. Boukerche, and A.A.F. Loureiro. Liaithon: A location-aware multipath video streaming scheme for urban vehicular networks. In *2012 IEEE Symposium on Computers and Communications (ISCC)*, pages 000436–000441, July 2012.

[261] Yang Wang and Greg Mori. A discriminative latent model of object classes and attributes. In *Proceedings of the 11th European Conference on Computer Vision: Part V*, ECCV'10, pages 155–168, Berlin, Heidelberg, 2010. Springer-Verlag.

[262] Qi Wei, Yinhao Ren, Rui Hou, Bibo Shi, Joseph Y. Lo, and Lawrence Carin. Anomaly detection for medical images based on a one-class classification. In Nicholas Petrick and Kensaku Mori, editors, *Medical Imaging 2018: Computer-Aided Diagnosis*, volume 10575, pages 375 – 380. International Society for Optics and Photonics, SPIE, 2018.

[263] S. Xie, T. Yang, Xiaoyu Wang, and Yuanqing Lin. Hyper-class augmented and regularized deep learning for fine-grained image classification. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2645–2654, June 2015.

[264] K. Yang, J. Ren, Y. Zhu, and W. Zhang. Active learning for wireless iot intrusion detection. *IEEE Wireless Communications*, 25(6):19–25, December 2018.

[265] L. Yang, P. Luo, C. C. Loy, and X. Tang. A large-scale car dataset for fine-grained categorization and verification. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3973–3981, 2015.

[266] L. Yang, P. Luo, C. C. Loy, and X. Tang. A large-scale car dataset for fine-grained categorization and verification. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3973–3981, June 2015.

[267] H. Yao, D. Fu, P. Zhang, M. Li, and Y. Liu. Msml: A novel multi-level semi-supervised machine learning framework for intrusion detection system. *IEEE Internet of Things Journal*, pages 1–1, 2018.

[268] Maram Bani Younes and Azzedine Boukerche. A performance evaluation of an efficient traffic congestion detection protocol (ECODE) for intelligent transportation systems. *Ad Hoc Networks*, 24:317–336, 2015.

[269] Maram Bani Younes and Azzedine Boukerche. Intelligent traffic light controlling algorithms using vehicular networks. *IEEE Trans. Veh. Technol.*, 65(8):5887–5899, 2016.

[270] Maram Bani Younes and Azzedine Boukerche. An efficient dynamic traffic light scheduling algorithm considering emergency vehicles for intelligent transportation systems. *Wirel. Networks*, 24(7):2451–2463, 2018.

[271] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. Adversarial examples: Attacks and defenses for deep learning. *IEEE Trans. Neural Networks Learn. Syst.*, 30(9):2805–2824, 2019.

[272] I. Zafar, E. A. Edirisinghe, and B. S. Acar. Localized contourlet features in vehicle make and model recognition. In *Proc. SPIE 7251, Image Processing: Machine Vision Applications II, 725105*, volume 7251, pages 725105–725105–9, 2009.

[273] I. Zafar, E. A. Edirisinghe, S. Acar, and H. E. Bez. Two-dimensional statistical linear discriminant analysis for real-time robust vehicle-type recognition. *Proc. SPIE, International Conference on Real-Time Image Processing*, 6496:649602–649602–8, 2007.

[274] B. Zhang, Y. Yu, and J. Li. Network intrusion detection based on stacked sparse autoencoder and binary tree ensemble method. In *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6, May 2018.

[275] Bailing Zhang. Reliable classification of vehicle types based on cascade classifier ensembles. *IEEE Transactions on Intelligent Transportation Systems*, 14(1):322–332, March 2013.

[276] J. Zhang, C. Chen, Y. Xiang, W. Zhou, and Y. Xiang. Internet traffic classification by aggregating correlated naive bayes predictions. *IEEE Transactions on Information Forensics and Security*, 8(1):5–15, Jan 2013.

[277] Zhenxia Zhang, Richard W. Pazzi, and Azzedine Boukerche. A mobility management scheme for wireless mesh networks based on a hybrid routing protocol. *Computer Networks*, 54(4):558 – 572, 2010. Advances in Wireless and Mobile Networks.

[278] Jun Zheng and Mingzeng Hu. An anomaly intrusion detection system based on vector quantization. *IEICE - Trans. Inf. Syst.*, E89-D(1):201–210, January 2006.

[279] Chong Zhou and Randy C. Paffenroth. Anomaly detection with robust deep autoencoders. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 665–674, New York, NY, USA, 2017. ACM.

[280] Husheng Zhou, Wei Li, Zelun Kong, Junfeng Guo, Yuqun Zhang, Bei Yu, Lingming Zhang, and Cong Liu. Deepbillboard: Systematic physical-world testing of autonomous driving systems. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, pages 347–358, 2020.

[281] Zhi-Hua Zhou. *Ensemble Learning*, pages 270–273. Springer US, Boston, MA, 2009.