Machine Learning-Based Decision Support to Secure Internet of Things Sensing

Zhiyan Chen

Thesis submitted to the University of Ottawa
in partial Fulfillment of the requirements for the
degree of Ph.D of Electrical and Computer Engineering

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

I hereby declare that I am the sole author of this thesis proposal. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Internet of Things (IoT) has weaknesses due to the vulnerabilities in the wireless medium and massively interconnected nodes that form an extensive attack surface for adversaries. It is essential to ensure security including IoT networks and applications. The thesis focus on three streams in IoT scenario, including fake task attack detection in Mobile Crowdsensing (MCS), blockchain technique-integrated system security and privacy protection in MCS, and network intrusion detection in IoT. In this thesis, to begin, in order to detect fake tasks in MCS with promising performance, a detailed analysis is provided by modeling a deep belief network (DBN) when the available sensory data is scarce for analysis. With oversampling to cope with the class imbalance challenge, a Principal Component Analysis (PCA) module is implemented prior to the DBN and weights of various features of sensing tasks are analyzed under varying inputs. Additionally, an ensemble learning-based solution is proposed for MCS platforms to mitigate illegitimate tasks. Meanwhile, a k-means-based classification is integrated with the proposed ensemble method to extract region-specific features as input to the machine learning-based fake task detection. A novel approach that is based on horizontal Federated Learning (FL) is proposed to identify fake tasks that contain a number of independent detection devices and an aggregation entity. Moreover, the submitted tasks are collected and managed conventionally by a centralized MCS platform. A centralized MCS platform is not safe enough to protect and prevent tampering sensing tasks since it confronts the single point of failure which reduces the effectiveness and robustness of MCS system. In order to address the centralized issue and identify fake tasks, a blockchain-based decentralized MCS is designed. Integration of blockchain into MCS enables a decentralized framework. The distributed nature of a blockchain chain prevents sensing tasks from being tampered. The blockchain network uses a Practical Byzantine Fault Tolerance (PBFT) consensus that can tolerate 1/3 faulty nodes, making the implemented MCS system robust and sturdy. Lastly, Machine Learning (ML)-based frameworks are widely investigated to identity attacks in IoT networks, namely Network Intrusion Detection System (NIDS). ML models perform divergent detection performance in each class, so it is challenging to select one ML model applicable to all classes prediction. With this in mind, an innovative ensemble learning framework is proposed, two ensemble learning approaches, including All Predict Wisest Decides (APWD) and Predictor Of the Lowest Cost (POLC), are proposed based on the training of numerous ML models. According to the individual model outcomes, a wise model performing the best detection performance (e.g., F1 score) or contributing the lowest cost is determined. Moreover, an innovated ML-based framework is introduced, combining NIDS and host-based intrusion detection system (HIDS). The presented framework eliminates NIDS restrictions via observing the entire traffic information in host resources (e.g., logs, files, folders).

Acknowledgements

I would like to express my sincere gratitude to my supervisor Dr. Burak Kantarci for his continuous support throughout my Ph.D. study. He opened the door for me to a novel research field, helped me to find a suitable research topic, and guided me successfully. The honest, careful and hard working spirit of Dr. Burak Kantarci will always inspire me. His timely advice, meticulous scrutiny, scholarly advice and scientific approach have helped me to a very extend to accomplish this task.

I am grateful to my colleagues in the SCVI-NEXTCON lab and friends, especially Dr. Murat Simsec, Jinxin Liu, and Ling Yi, who provided me with their time, support, and encouragement, particularly during challenging times.

Finally, my sincere thanks also goes to my husband Dongming, my daughter Jiaxin, and my son Jiaxu, for their unwavering love and support, and for always believing in me. Their constant encouragement and sacrifices made it possible for me to pursue my academic goals

Without the support of these individuals and organizations, this thesis would not have been possible. Thank you.

Table of Contents

List of Tables

List of Figures

Glossaries

$3GPP$	: 3rd Generation Partnership Project
$ADASYN$	: Adaptive Synthetic over-sampling
AI	: Artificial Intelligence
$APWD$	: All Predict Wisest Decides
AS	: Attack Severity
BLE	: Bluetooth Low Energy
CNN	: Convolutional Neural Networks
DBN	: Deep Belief Network
DFD	: Data Flow Diagram
DoS	: Denial of Service
DR	: Detection Rate
DT	: Decision Tree
FL	: Federated Learning
FN	: False Negative
FP	: False Positive
FPR	: False Positive Rate
GAN	: Generative Adversarial Network
IoT	: Internet of Things
MCS	: Mobile Crowdsensing

ML	: Machine Learning
$NIDS$	: Network Intrusion Detection System
NB	: Naive Bayes
NFT	: Non-fault Tolerance
$OADR$	: Original Attack Detection Rate
$PBFT$	: Practical Byzantine Fault Tolerance
PCA	: Principal Component Analysis
$POLC$	: Prediction of the Lowest Cost
PoW	: Proof of Work
PoS	: Proof of Stake
RAA	: Recall After Attack
RBA	: Recall Before Attack
RBM	: Restricted Boltzman Machines
RF	: Random Forest
RNN	: Random Neural Network
$RoFT$	: Ratio of Fake Tasks
$RoLT$	: Ratio of Legitimate Tasks
$SOFM$	: Self Organizing Feature Map
TP	: True Positive
TN	: True Negative
WSN	: Wireless Sensor Network

Notation and symbol

p	: Fault probability for each node in the blockchain
N	: Total number of nodes in the blockchain
k	: Number of faulty nodes tolerated by PBFT to reach consensus
t	: Number of active nodes in blockchain
p_{ft}	: Probability for achieving a blockchain make consensus under PBFT
p_{nf}	: Probability for achieving a blockchain make consensus without fault tolerance
p_c	: Probability for a centralized system deciding a transaction if saving
w	: Number of legitimate tasks with correct prediction
v	: Number of fake tasks with incorrect prediction
T_f	: Number of fake tasks in test dataset
T_l	: Number of legitimate tasks in test dataset
$RoFT$	: Ratio of Fake Tasks
$RoLT$	: Ratio of Legitimate Tasks
$NFTSystem$	: Non-fault tolerant MCS system
$CLSystem$	: Centralized (CL) MCS platform
MLi_Pj	: F1 score of ML model i for class j
MLx	: ML model with the maximum $MLi\_F1\_max\_num$ value
MLy	: ML model with the maximum $MLi\_F1\_max\_num$ value after MLx
SPx	: Set of classes where F1 score under MLx ranks the first

SPy	: Set of classes where F1 score under MLy ranks the first
SPa	: $SPa = MLx \cup MLy$, union of set MLx and MLy
n	: Total number of classes in a dataset
N	: Total number of instances in a test dataset
m	: Total number of ML models
X_k	: The kth instance in a test dataset
$R_{MLx}^{X_k}$	: Prediction value of MLx for the kth instance
$R_{MLy}^{X_k}$	: Prediction value of MLy for the kth instance
R^{X_k}	: Aggregation model decision for the kth instance=

Publications

- Conference
  - Chen, Zhiyan, et al. "Deep belief network-based fake task mitigation for mobile crowdsensing under data scarcity." ICC 2020-2020 IEEE International Conference on Communications (ICC). IEEE, 2020.
  - Chen, Zhiyan, Murat Simsek, and Burak Kantarci. "Region-aware bagging and deep learning-based fake task detection in mobile crowdsensing platforms." GLOBECOM 2020-2020 IEEE Global Communications Conference. IEEE, 2020.
  - Chen, Zhiyan, et al. "All Predict Wisest Decides: A Novel Ensemble Method to Detect Intrusive Traffic in IoT Networks." 2021 IEEE Global Communications Conference (GLOBECOM). IEEE, 2021. (**BEST PAPER AWARD**)
  - Chen, Zhiyan, Murat Simsek, and Burak Kantarci. "Federated Learning-Based Risk-Aware Decision to Mitigate Fake Task Impacts on Crowdsensing Platforms." IEEE International Conference on Communications (ICC). IEEE, 2021.
  - Chen, Zhiyan, and Burak Kantarci. "Adversarial Machine Learning-Driven Fake Task Anticipation in Mobile Crowdsensing Systems." 2021 IEEE International Conference on Service-Oriented System Engineering (SOSE). IEEE, 2021.
  - Chen, Zhiyan, and Burak Kantarci. "Generative Adversarial Network-Driven Detection of Adversarial Tasks in Mobile Crowdsensing." [Accepted by IEEE ICC 2022].
- Journals
  - Chen, Zhiyan, Claudio Fiandrino, and Burak Kantarci. "On blockchain integration into mobile crowdsensing via smart embedded devices: A comprehensive survey." Journal of Systems Architecture 115 (2021): 102011.
  - Chen, Zhiyan, et al. "Machine Learning-Enabled IoT Security: Open Issues and Challenges Under Advanced Persistent Threats." ACM Computing Surveys (CSUR) (2022)
  - Chen, Zhiyan, Omer Melih Gul, and Burak Kantarci. "Practical Byzantine Fault Tolerance-based Robustness for Mobile Crowdsensing." Distributed Ledger Technologies: Research and Practice (2023)

  - Zhiyan Chen, Murat simsek, Burak Kantarci, Petar Djukic, All Predict Cost Efficient Decides: A new Cost-Centric Ensemble Learning for the Detection of Network Intrusions, ACM Digital Threats Research and Practice(under review)
  - Zhiyan Chen, Murat simsek, Burak Kantarci, Mehran Bagheri, Petar Djukic, Machine Learning-Enabled Hybrid Intrusion Detection System with Host Data Transformation and an Advanced Two-Stage Classifier (under review)

- Patents
  - Zhiyan Chen, Murat Simsek, Burak Kantarci, Petar Djukic, James P'ford't Carnes III, Mehran Bagheri, Jinxin Liu, Yu Shen, Machine learning detection of network attacks using traffic and log information, utility patent
  - Chen, Zhiyan, et al. "Machine Learning-Enabled IoT Security: Open IChen, Zhiyan, Murat Simsek, Burak Kantarci, Petar Djukic. "All Predict Wisest Decides: A Novel Ensemble Method to Detect Intrusive Traffic in IoT Networks.", provisional patent

Chapter 1

Introduction

In recent times, new technologies have enabled connection across diverse devices simpler than ever before, thanks to considerable improvements in networking devices and computer systems. Roughly more than 25 billion devices are predicted to be linked to the Internet, according to different estimates [156]. As a result, the founded notion of the Internet of Things (IoT) has emerged [156]. The IoT is a network of devices equipped with sensing, communicating, and unique identifiable characteristics [108]. The widespread adoption of connected services that utilize IoT devices and networks has enabled IoT applications' advancement. IoT networks offer smarter environments and services than earlier networks by leveraging and integrating sensed data from various sensors [108]. Embedded sensors in IoT devices communicate with peer nodes through wireless networks such as the standards developed by the 3rd Generation Partnership Project (3GPP), IEEE 802.11ah, Bluetooth Low Energy (BLE), and Z-Wave protocols [4]. Additional benefits include transparency and ease of access to information while reducing human interventions. In respect to applications, IoT networks have been integrated with various domains such as smart city [209], industrial processes [189] and intelligent transport systems [269] and e-health [25]. Due to a number of advantages of IoT, such as improved customer engagement, reduction of waste, enhanced data collection, and technology optimization, IoT technology is being integrated into the various system (e.g., industrial control systems, modern vehicles, and critical infrastructure) [204]. For instance, the integration of IoT systems with critical infrastructures such as facilities, power distribution networks, and critical assets is emergent [23]. One such example is the use of IoT systems to monitor and manage critical infrastructures in the smart city context [231]. Such networks' objective is to improve efficiency in management, production, and services of a city to make the city more livable.

However, it is widely recognized that IoT technologies and applications are still in their

infancy [54]. Several challenges restrain the development and application of IoT, including technical limitations, lack of standardization, and limited security and privacy components [140]. The acceptance and ubiquity of new IoT technologies and services largely depend on the status of data security and privacy protection, which are two challenging issues for IoT because of its deployment, mobility, and complexity [55]. Machine learning (ML)-based approaches are widely deployed to secure IoT systems. In this work, we aim to identify network intrusions in IoT systems via ML-based methods. IoT has been an emerging paradigm to interconnect a plethora of sensors and uniquely addressable devices to exchange data directly without human intervention [117, 19]. The essential purpose of IoT is to contribute a network framework with general protocols and software to enable connecting and incorporating of all connected nodes in the network[8, 55]. For instance, an IoT-based smart home consists of many devices (e.g., smart appliances, smart locks, smart hubs) to collect and monitor various data types in our daily life [221]. Due to their large-scale deployment and ubiquity, IoT systems confront critical cybersecurity threats, among which network intrusion appears to be one of the top vulnerabilities as a massive number of connected devices lead to a giant attack surface in IoT systems [116]. In order to address network intrusion, network intrusion detection systems (NIDS) are typically deployed in IoT systems. A NIDS is used to detect potential perspective attacks (e.g., malicious operations, various attacks) and notify the appropriate persons based on detection results. ML-integrated approaches are widely deployed and proven the effectiveness to identify attacks in IoT networks. For instance, the study in [216] introduces Gaussian Mixture clustering with Random Forest Classifiers as well as using K-means clustering with Random Forest Classifiers in order to detect intrusions. The authors report that this proposed hybrid approach decreased the false alarm rate to 0.04%. Although ML algorithms are widely deployed against network intrusion, their performance on identifying each intrusion type varies. Thus, under the same test dataset, it is possible to observe an ML model (e.g., model_1) perform better for an attack-type (e.g., class_1), and another model (e.g., model_2) outperform the former for other classes (e.g., class_2 and class_3). It is not straightforward to use one ML model as a one-size-fits-all approach. An ML model demonstrates expertise in predicting a specific set of classes. Detection performance could be boosted if taking advantages from various ML models as a result to contributing a more wiser prediction value. Based on this motivation, we propose an innovative ensemble framework, namely All Predict Wisest Decides (APWD) method and Prediction of the Lowest Cost (POLC), that are general methods that can be used as benchmark methodology to integrate various ML models. Moreover, host features are utilising for intrusion detection, that is helpful to identify intrusions [165] [202]. Network intrusion detection is discussed in detail in Chapter 5.

Meanwhile, as an integral part of IoT, Mobile Crowdsensing (MCS) via smart embedded

devices has recently gained significant attention for being effective in a wide range of applications and services, particularly in smart environments. MCS is a new research tendency in the IoT that connects consumers with smart mobile devices. MCS is growing rapidly as a decentralized concept that lies at the crossroads of IoT and contributing sensing data voluntarily / limited profits. MCS invents a new way of seeing the world in order to considerably expand the IoT service and investigate a new generation of networks that link device to device, user to user, and device to user [144].

MCS is a technique where a large group of users utilising smart devices such as smart phones, wearables, tablets and in-vehicle sensing devices equipped with various sensors to share sensory data from surroundings [37]. With the motivation of potential benefits to collect data without additional costs, MCS is extensively researched by scholars and has various applications such as vehicles Crowdsensing paradigm [128], noise monitoring [95], extreme driving behaviors detecting [82] and so on. However, MCS development confronts several challenges such as privacy and security, data quality, participant recruitment and so on [47]. Thanks to the studies that ensure differential privacy in the recruitment of participants in MCS campaigns [276], there has been significant progress in MCS security from the standpoint of participant privacy. Since MCS platforms have no control over the recruited participants, truth discovery in the presence of data poisoning attacks have been other security measures that have been being investigated to secure the MCS systems [95, 267]. Indeed, the threats introduced by data poisoning attacks also require sustaining the MCS platforms by eliminating participants that are not trustworthy or dependable [109, 196, 197].

A grand challenge in MCS is not only the participation of untrusted parties but the vulnerable nature of this distributed sensing system due to openness for any type of task and sensory data submissions without prior knowledge about the submitter. Thus, identifying and eliminating fake sensed data and fake sensing tasks is of paramount importance for the performance of MCS platforms. The literature includes game theoretic solutions against fake sensed data [197, 250]. Meanwhile, especially energy limitation is another major challenges. MCS activities consume smart devices resources which are qualified in terms of energy, bandwidth and computation capability [73]. To tackle this issue, [253] presents an innovative MCS framework to reduce energy consumption. In addition, cloud-assisted collaborative sensing methodology is applied to decrease devices' power consumption in MCS activities [213]. The study in [125] introduces a novel system Piggyback crowdsensing system to collect sensing data from smart phones under awareness of reducing energy consumption. Besides normal energy consumption in MCS platforms, attackers may inject malicious tasks to MCS servers with the aim of consuming more resources including power from users' devices. These are also addressed by different artificial intelligent-based

methods so to detect and mitigate illegitimate tasks [272, 274, 275, 51]. It is worth to note that fake sensing task submissions pose security threats for both the MCS platforms by clogging sensing servers and participating devices by draining their batteries and other resources for invalid tasks / requests. Most recently in [218], a feasibility study of deep learning networks have been presented for the detection of fake task submissions.

MCS activities are initiated by task requesters who submit sensing tasks to the MCS platform. A sensing task consists of distribution location, time, battery requirement, and other sensitive data which cybercriminals could steal to speculate requesters' personal information. Therefore, preventing sensing task information from being stolen aims to improve the adoption of MCS applications. Furthermore, sensing tasks could be modified unintentionally by an authorized person or malicious users. It is critical to ensure tamper-proof sensing tasks in MCS systems. Meanwhile, malicious requesters attack the MCS platform by submitting fake sensing tasks to clog MCS platforms and drain more energy from participants' devices [272]. We should pay attention to protecting sensing task information and eliminating malicious tasks before distributing tasks to participants.

Blockchain is intensively deployed in the MCS system to preserve participants' privacy and security [277, 191]. Thanks to its decentralized, immutable, and traceable properties, blockchain protects users' data, and information [47]. An MCS platform is responsible for collecting sensing tasks and distributing them to potential participants, generally designed as a centralized entity. However, a centralized MCS platform confronts several challenges such as the single point of failure problem, reliability in workers' recruitment, unbiased evaluation for awards distribution, and possible breach of users' sensitive information[104]. Blockchain is a powerful technique to implement a decentralized MCS platform, avoid sensing tasks tampering, and against requesters' privacy breach in MCS [71]. The study in [47] surveys blockchain-enabled MCS framework which encourages participation, preserves security and privacy, ensures data quality.

A consensus protocol is the backbone of a blockchain network that guarantees all transactions verified and secure to be added without a centralized authority. Proof of Work (PoW) is a widely known consensus protocol that has been adopted in the context of cryptocurrencies. However, it suffers from extremely high computational resource demands and overlong latencies (e.g., approximate 10 minutes in Bitcoin and 15 seconds in Ethereum) [132]. Proof of Stake (PoS) is another renowned consensus protocol that aims to reduce the computational overhead. Despite this motivation, PoS requires an overwhelming amount of resources, which may prolong the transaction approval delay up to 6 minutes as pointed out in a PoS-based protocol, Chains-of-Activities [27]. Recently, Practical Byzantine Fault Tolerance (PBFT) consensus protocol gained people's attention [134]. PBFT protocol can tolerate up to 1/3 faulty nodes to reach consensus. Moreover, PBFT is a lightweight con-

sensus mechanism that does not require many measuring resources for reaching consensus. Therefore, PBFT is convenient for an IoT system consisting of many resource-constrained devices. PBFT decreases implementation complexity, resulting in less effort applying it in a blockchain. Furthermore, PBFT has less energy consumption during the consensus process than prominent consensus approaches, PoW and PoS [32]. The study in [81] introduces the prevailing consensus protocols and shows ideas to revisit traditional Byzantine consensus in the context of blockchain.

A blockchain using PBFT consensus protocol is designed to implement a distributed decentralized MCS system in this work which prevents the single point of failure and avoids sensing task tampering. PBFT-powered blockchain is applicable for MCS systems depending on resource-constrained devices (e.g., smartphones, wearable equipment) to submit sensing data [132]. Meanwhile, with the motivation to identify fake task attacks [272], ML algorithms are deployed in several blocks in the blockchain. An ensemble learning comprises these ML models in blocks and a center server entity, Aggregator. ML models in blocks train and predict independently and prediction results are not shared with other ML models. The Aggregator model collects individual ML algorithm prediction results and makes a final decision following majority vote rule. Since the aggregator is a centralized entity, it introduces a single point of failure issue in fake task detection. Thus, if the Aggregator fails, elimination of illegitimate tasks gets disabled. However, this issue can be addressed via survivability design of the proposed architecture. According to a survivable design, a robust replication mechanism can ensure resiliency of the system in the case of the failure of the aggregator. The aggregator entity is part of an ensemble learning which aims to boost fake task detection performance. Therefore, an optimization model to ensure survivable design of the proposed system against possible failure of the aggregator is essential to maintain the decentralized nature while boosting the fake task elimination performance. We leave the development of such optimization model to our immediate future work. The proposed MCS framework consists of the requester, blockchain network-based MCS platform with an Aggregator, and participants. The blockchain-integrated MCS platform is the core entity in this framework which ensures a decentralized MCS platform, secures the MCS system against fake tasks, and prevents sensing task tampering.

1.1 Problem Statement

According to previous introduction of IoT systems, there are still several challenges and problems that need pay attentions.

- In MCS scenario, the previous studies assumed high availability of sensed data but did not consider the challenges, that limits MCS-based application development. Specifically, in current research, it rarely discusses sensing data under the conditions of data scarcity and sensitive of illegitimate tasks submission regions. It is necessary to address these issues to boost attack detection performance further to secure systems.

- Traditional MCS platform consists of a centralized platform, that leads vulnerabilities for in MCS system such as the single point of failure and potential data breach. It is important to build a decentralized MCS system instead of conventional one and prevent data tempering and breaching in MCS activities.

- IoT devices generate large volumes of data, which makes it difficult to analyze and detect anomalies that may indicate an intrusion. ML-based NIDS is widely used to observer and analyze high volume data generated in IoT networks due to ML automatically learn patterns and relationships from the data without being explicitly programmed and finally identifying intrusions. However, ML-based NIDS becomes difficult to achieve promising performance. On one hand, it is problematic to select a suitable ML model applicable to all classes prediction due to ML models performing divergent detection performance in each class. On the other hand, NIDS identify intrusion via observing real network traffic and analyzing suspicious activities. It results in limitations to detect specific types of attacks, such as Advanced Persistent Threats (APT).

1.2 Contributions

The thesis aims to bridge these challenges in IoT-related streams, including fake task detection in MCS and network intrusion detection in IoT, so the contribution of this thesis includes:

- **Boosting fake task detection performance:** Several mechanisms are introduced to boost fake tasks detection performance, including addressing fake tasks scarcity via several oversampling techniques, extracting more features via unsupervised ML algorithms, and designing FL-based decentralized systems.

- **Blockchain-integrated MCS to protect data:** A blockchain-driven MCS framework is proposed to address the centralized MCS system vulnerabilities such as the

single point of failure and sensing task data tampering. Moreover, the blockchain network combines Practical Byzantine Fault Tolerance (PBFT) consensus algorithm that provides a robust MCS framework tolerating faulty nodes, making the MCS system robust and vigorous.

- **NIDS and HIDS to detect network intrusions:** In IoT systems, it is critical to identify network intrusions to prevent these attacks and secure entire system. We firstly introduce a generic ensemble-based NIDS systems to detect intrusive traffic in IoT networks which takes advantages (e.g., detection performance or cost) from different ML algorithms to make decision. The proposed approach improves attack detection performance critically when compared with conventional approaches. Meanwhile, NIDS and HIDS hybrid framework is proposed to improve intrusion detection performance and accuracy.

Chapter 2

Related Work

IoT confronts several challenges restrain the development and application and security is one of the most critical issues [140]. As an important part of IoT, MCS system security has been investigated in recent literature including privacy protection, fake sensing data, and fake sensing tasks. Meanwhile, several techniques and mechanisms are studied that are essential parts of ML-based approaches (e.g., oversampling techniques, feature extraction methods, and decentralized MCS systems). Meanwhile, network intrusion detection has become a paradigm and various papers have demonstrated their ideas and approaches in recent years. ML-based and deep learning-based methods have been proposed in current literature to deploy network intrusion detection system, especially ensemble learning-integrated approaches, which demonstrate promising detection performance. Therefore, this chapter introduces the state-of-arts to address issues in MCS scenarios (e.g., generating extra samples for minority classes, extracting more features in original dataset, deploying decentralized systems) to ensure MCS system security, intelligent attacks generating by ML-based mechanisms, and network intrusion detection via ML-integrated approaches.

2.1 MCS System Security

MCS is invested extensively in the context of security preservation. Non-dedicated property of MCS platform results in security and privacy issues, such as false data submissions, fake task attacks, user location leaking and eavesdropping [47]. Artificial Intelligence (AI) methods including Deep learning and ML methods have been shown to be the most effective method to detect and mitigate intrusion in digital systems. In traditional MCS campaigns, an MCS platform requires exact location information of participants to assign

tasks optimally, raising privacy concerns from participants' side [31]. In order to eliminate location privacy vulnerability to adopt MCS-based services widely, there are numerous approaches introduced. The study in [242] proposes an e-difference privacy policy used in a sparse MCS system. This approach provides a theoretical guarantee for protecting participants' location privacy without considering the opponent's prior knowledge. The methodology builds on three procedures: 1) applying a small change for crowd-sensed data to map original data to new data with an obfuscated location, 2) based on linear programming, determining the optimal position fuzzy function under the objective to minimize the confusion of data modification, 3) enhancing interpretation accuracy of obscure data by an uncertain perceptual fuzzy formula. In [262], the authors aim to authorize anonymously authenticated blind signature, allowing mobile users to register at particular times. Based on the concept of blind signature, the authors demonstrate an analogical approach of anonymous credit for mobile users. Therefore, the MCS platform refers to an anonymous reputation when assigning tasks to workers. Meanwhile, in variance with the quality of crowd-sensed data submitted by participants, the MCS platform adjusts to data contributors' anonymous reputations. A framework for MCS task selection is proposed in [258] on the premise of ensuring location privacy, reducing resource consumption, and assuring task profitability. The framework describes how participants can customize their privacy requirements to prevent exposure of precise location. In [257], based on ensuring location privacy, the authors conduct task bidding and distribution under two task selection targets: 1) decreasing the total cost to lowest, and 2) decreasing the average cost to lowest. A participant submits the cost of fuzzy processing by applying a differential privacy method. The MCS platform determines the winner claiming low cost and determines the winner's reward according to a probabilistic personal rational approach. The study in [212] introduces a privacy-preserving task distribution system that aims to optimize task acceptance rate and preserves participants' location information replying on edge nodes. Edge nodes are used to replace a user's accurate location data with an obfuscated location to protect private data. It means edge nodes are agents for task allocation that avoid a crowdsensing server without authorities to access a participant's location data.

MCS is vulnerable to fake sensing data and data poisoning attacks. Fake sensing data means a number of malicious participants either cooperate or act individually to submit false information resulting in disinformation on an MCS platform. For example, a group of selfish users upload fake crowd-sensed data to MCS servers to reduce sensing costs and protect their sensitive information. Meanwhile, an MCS system is exposed to data poisoning attacks, similar to fake sensing data attacks. Data poisoning attacks are performed by malicious participants to submit false sensing data that decreases the effectiveness of the MCS scheme [130]. Data poisoning leads to inaccurate results of data

analysis via injecting incorrect data. Fake sensing data and data poisoning attacks mean participants upload incorrect or false data to the MCS platform. It is different from sensing tasks being affected, which shows requesters submitting malicious tasks to the platform. With the current availability to use truth discovery algorithms, data poisoning attacks are mitigated significantly [278]. However, data poisoning attacks influence the TruthFinder's framework effectiveness, especially when the malicious users have illegal authority to view MCS information. In [130], data poisoning attacks are investigated in MCS, and attackers only have local information authority level. However, malicious users have developed the capability to inject poisoning data that reduces the effectiveness of TruthFinder-based MCS platforms. The authors in [159] study the availability of data poisoning attacks and mitigation approaches in MCS through authorizing a truth determination mechanism. They implement an optimal attack system where malicious users can maximize attacks and pretend to be regular participants, thus requiring a robust cybersecurity response. In [281], a novel framework is presented to enhance MCS data's credibility by the mitigation of collusion threats, which means several malicious users cooperatively submit incorrect data to misguide the MCS system. The platform measures remote crowd-sensed data's spatial correlation and determines the correlation between remote crowd-sensed data and local sensing data. The framework boosts the accuracy of error detection and the overall data trustworthiness. The authors in [252] show a game-based approach to identify faked sensing attacks. Specifically, an MCS server deploys a deep reinforcement learning-based identification approach to evaluate crowd-sensed data accuracy and predict participants' effort for sensing data collection. The MCS platform penalizes participants who contribute to incorrect data to prevent fake sensing data uploading. The authors in [96] introduce an approach to detect data poisoning attacks via estimating resource consumption and filtering data before aggregation.

Due to the openness of MCS platforms, it is challenging to deploy effective assessment mechanisms to prevent malicious requesters from submitting illegitimate tasks to MCS platform. Hence, an MCS platform is vulnerable to attackers aiming to decrease the platform's effectiveness and suppress users' willingness to participate in MCS tasks. In this thesis, fake task attacks are investigated extensively since they are harmful for MCS platform and participants. Fakes task attack are one of the most critical issues in an MCS system. Machine learning-based approaches are investigated for fake task mitigation in recently years. For instance, the studies in [273] describe illegitimate tasks aiming to drain more energy from users' equipment and/or clog the sensing servers. On the one hand, completing crowdsensing tasks consume batteries, bandwidth, and storage from participants' devices; therefore, smart devices' resource limitations lead to MCS systems vulnerabilities that can be roadblocks in achieving high participation MCS campaigns [38]. On the

other hand, distributed denial of service (DDoS) attacks in MCS's context declines service capability. It reduces participation by injecting illegitimate energy-oriented and location-oriented tasks into the MCS platform. To tackle the significant security issue affecting both participants and the platform, the authors in [273] apply an artificial intelligence-based approach. They deploy machine learning-based methods in an MCS platform server to detect fake tasks before publishing them to participants. The authors in [270] design a self-organizing feature map-based attack model to block an MCS server that is used to affect most participants considering the location of attack zones. The zones are determined through classifying 2-D coordinates of all potential participants and identifying any other smart devices in the area. In [217], several deep learning-based methods for fake task mitigation are investigated, such as deep autoencoder, Restricted Boltzmann machine, and Deep belief network. In [24], it was reported that smart spaces could be protected against intrusive patterns by bridging ML with intelligence driven by data. Indeed, MCS exhibits its unique vulnerabilities due to leveraging ubiquity, heterogeneity and participation of mobile smart devices in sensing tasks [112]. As an example, the authors in [160] study ML-based design of data poisoning attacks in MCS platforms. Furthermore, the study in [80] proposes to use ensemble classification algorithms to overcome ML-driven attacks that aim to extract contextual information. In [272], Zhang and Kantarci, and later in [274] Zhang et al., proposed to use ML against fake task submissions in MCS. Moreover, state of the art techniques (e.g., blockchain) are applied in order to protect MCS platforms. For instance, the study in [18] proposes a Blockchain-based framework to detect and prevent fake sensing activities in MCS. The study in [133] presents an innovated MCS framework to avoid eavesdropping by integrating friendly jammers to MCS system. Indeed, AI methods are proven to be effective approaches to address MCS security and privacy issues. For instance, the authors in [24] present ML to protect smart spaces against intrusive patterns. The study in [160] proposes to prevent poisoning attacks by a ML-based methodology in MCS platforms. Furthermore, the study in [80] demonstrates ensemble learning-based methods to tackle privacy issue with motivation to extract contextual information in MCS. The authors in [272, 274] propose an ensemble learning-based classification method to detect fake tasks in MCS. As for the studies that leverage deep learning to secure MCS platforms, the works in [249, 218] are the example works that investigate the possibility of deep learning networks.

Although ML-based approaches are introduced in various literature, some specific issues in MCS system security research have not been investigated yet, such as limited samples in dataset, without enough features, and issues from centralized MCS systems. Imbalanced dataset is common in real world problems such as failure diagnosis, spam detection, anomaly detection, fraud detection and medical diagnosis, which affects the quality of deep

learning and machine learning. Therefore, to tackle imbalanced dataset problem, authors in [264] propose DBN based re-sampling support vector machine ensemble learning paradigm to solve the imbalanced data problem. As well as cost-sensitive Deep Belief Networks is demonstrated for imbalanced classification problems [266]. Besides the scarcity of the data regarding sensing task submissions, the scarce data is also imbalanced. There are two directions to balance the dataset: enlarge the minority dataset through over-sampling; reduce the majority data through under-sampling. Since we consider the condition of scarcity of sensing task submissions in the MCS platform, it is viable to proceed with oversampling. The Synthetic Minority Over-sampling Technique (SMOTE) [152], Borderline SMOTE [84], Random over-sampling [164] and Adaptive Synthetic over-sampling (ADASYN) [83] are the widely known and common over-sampling methods , which are adopted and tested in our proposal. All of these oversampling methods differ from each other with their insertion methods. The SMOTE aims to find the nearest neighbors for each minority class data followed by the generation of random points to the nearest neighbors of the sample. ADASYN can be considered a similar scheme to SMOTE with minor improvements as it adds a small random value to the data points. Borderline SMOTE is a variant of the SMOTE algorithm where a minority class borderline aims to be detected and considered to generate new synthetic samples. Lastly, the Random over-sampling method enlarges the minority class by picking samples of minority classes and reusing them multiple times. It is indeed essential to choose the most suitable over-sampling method for the highest possible performance of the DBN. Meanwhile, Generative adversarial networks (GANs) [14] provide a more intelligent way to augment data samples other than the traditional oversampling methods (e.g., random sampling [167], synthetic minority oversampling technique (SMOTE) [43]). A GAN's objective is to create synthetic samples which are as comparable to original samples as possible using neural networks.

Data preprocessing is to process features of data so that algorithms can explain it simply. As stated in [98], preprocessing data by extracting features from the original dataset boosts neural network-based classifier performance. In this paper, we extract an additional region feature upon an attack zone by k-means algorithm [118] and deploy prevalent and standard preprocessing approaches including Principle Component Analysis (PCA) [58] for dimensionality reduction, Borderline SMOTE over-sampling method [85] to tackle imbalanced dataset problem and standardization. These preprocesing methods are common and widely used. Authors in paper [194] compare several data preprocessing methods including resizing, face detection, cropping, adding noises, and data normalization consists of local normalization, global contrast normalization and histogram equalization and show the influences of its in CNN performance.

It is normal to observe that MCS systems are centralized. It leads to a centralized de-

fense system to prevent attacks in MCS system. However, a centralized system contributes several disadvantages such as privacy information leakage, low data quality, and low participation rates [47]. The authors in [119] demonstrate a framework to prevent potential leakage of mobile users' sensitive information using a cloud proxy. The location privacy methodology blurs the participants' location, which encourages participants to involve sensing tasks. A quadtree generated by the cloud agent enables a direct connection between the requester and participants in MCS. Compared to centralized mechanisms, this approach claims decentralized storage and maintenance of quadtree structures. The authors in [33] develop a participatory sensing system targeting mobile phones with the Android system. Meanwhile, they show a powerful and convenient privacy protection approach since participants need not provide sensitive information. The method is essentially based on the assumption of a smartphone generating and storing data by itself rather than submitting data to a centralized management entity. While data is traceable and immutable in a blockchain-based MCS platform, blockchain-based MCS systems overcome various vulnerabilities and improve MCS platform in various ways such as improving worker reliability, introducing a fair evaluation of the rewarding procedure, preserving sensitive information, and reducing deployment costs [93, 70]. FL-based approaches are introduced to address issues in MCS settings as FL is a promising AI technique that trains an algorithm across multiple decentralized edge devices or servers holding local data samples, without exchanging them [260]. The studies in [106, 245] aim to determine trusted and respectable users in FL tasks according to reputation metrics. The authors in [169] demonstrate a FL-backed approach to identify jamming attacks in flying ad-hoc networks and achieve a promising detection performance. A FL-based approach is introduced in [265] to have edge units contribute to the train of the models. Meanwhile, the study in [237] relies on Convolutional neural network (CNN) model with the motivation of improving the detecting rate and precision of detection against malware attacks.

2.2 Blockchain integrated MCS

Blockchain emerges as a decentralized system with traceability and tampering resistance manners, overcoming IoT systems vulnerabilities. Many recent works combine blockchain with IoT systems. The survey [66] identifies the adoption of blockchain into IoT-based applications in engineering. The authors in [126] presented a private blockchain-based framework aiming to protect and trustworthy industrialized activities, which was used to manage to access to paramount sensors and the collecting data. Researchers have paid significant attention to adopting blockchain techniques in MCS schemes in recent years. The

study in [256] presents a decentralized MCS system integrating blockchain that includes an incentive mechanism Utility-Oriented Role-Centric Incentive Mechanism (URIM), to boost participants' utility. It reduces computing requirements and ensures single entity rationale and reliability. The study in [191] proposes a blockchain technique in an MCS framework to preserve participants' privacy by implementing a distributed MCS system. The studies in [94, 228, 248, 74] introduced a blockchain-based MCS to address the single point of failure issue in the conventional centralized MCS systems. Specifically, the authors in [94] relied on miners in blockchain for data quality verification, and they implemented a scheme to detect suspicious sensing data and eliminate it to ensure data quality. The study in [277] presented an MCS framework integrating blockchain and edge computing to guarantee reliability and effectiveness, which protects user privacy via managing their reputation. Meanwhile, the authors developed a prototype system for evaluation that demonstrated a higher utility and more secure in detecting malicious participants. In [93], blockchain was applied in the MCS framework to preserve privacy and security in both submission procedures of sensing data and participants' incentive procedure.

The authors in [228] introduce ChainSensing, a decentralized MCS architecture that leverages blockchain. In ChainSensing, mobile users connect with the blockchain via smart contracts to carry out their activities, such as posting sensing tasks and reporting acquired data. Alongside this, a heuristic approach is designed to address the path planning problem via computational prophets and embedded into users' smart equipment. The study in [74] presents TrustWorker, a worker recruitment strategy for blockchain-based crowdsensing to improve reliability and participant privacy. The TrustWorker inherits the benefits of blockchains (i.e., decentralization, transparency, and immutability) with the aim of improving the reliability of the participant recruitment procedure. To safeguard the reputation information of each participant, the authors encrypt reputation values with a deterministic encryption process and then choose the top participants using a confidential least representative approach. In [248], on the basis of the dual blockchain network, the authors provide an anonymous reputation monitoring system in which reputation values are hidden. Specifically, one blockchain is utilized to maintain and refresh reputation ratings, while another blockchain publishes sensing tasks and logs task information. In smart contracts, a type of ring signature and Pedersen commitment is used to exchange and validate reputation ratings ina confidential manner without compromising their uses in the data processing stage.

Researchers have worked on the integration of blockchain to identify attacks, in order to preserve sensitive data and identify threats; ensemble learning-integrated framework could assist in the detection of complicated harmful events while also ensuring data privacy; additionally, these approaches could be utilized for extra protection guarantees in the cloud,

as well as to safeguard IoT networks [13]. An ensemble learning model provides a solution that involves coordinating many mobile devices to train a standard weak learner independently [282]. The study in jia2020anti demonstrates a decentralized system integrating ensemble learning with blockchain to detect DDoS attacks. The proposed system leverages Adaboost, RF and several lightweight ML models (e.g., CART, ID3) in the ensemble algorithm. The authors in [100] introduce the detection method in the blockchain environment has significantly improved generality, and universality, allowing it to reliably recognize the onslaught aspects of DDoS attacks in peer to peer networks. In [261], a novel framework is built combining blockchain and ensemble learning for characterization of anomalies in wireless sensor networks (WSNs). By establishing an appropriate framework and consensus method, the overall classification method may be updated repeatedly to improve detection accuracy. Moreover, the blockchain ensures trustworthiness of networks, enabling the identification system immune to internal threats. Lastly, the evaluation results show that the proposed method outperforms other comparable systems in terms of performance and cost. The study in [142] demonstrates a framework to integrate blockchain and an ensemble learning for network intrusion detection with vehicular edge computing, and tackles the challenges of conventional intrusion detection systems that require advanced computing resources to train ML algorithms. Blockchain is designed to store and share the training model, which secures the aggregation procedure in the ensemble algorithm. The authors in [89] propose a cloud-based intrusion detection technique that combines blockchain and ensemble model that transfers the IoT intrusion alarm in local devices to a cloud platform for global prediction. Moreover, the local model training process information and behavioral data are stored in the blockchain.

Four entities comprise a blockchain-integrated MCS system, namely *miner*, *agent*, *requester*, and *worker*. *Miners*, who also can be requesters or workers, are responsible for deciding if a new transaction is acceptable and, if so, saves it in a global ledger. Since miners provide computing resources, they typically are paid a transaction fee. A *requester* initiates and submits a task approved by a worker to acquire certain information. *Agents* are associate miners between public blockchains and private blockchains. An agent is responsible for transferring tasks in the public blockchain network to a private blockchain network. Participants who are willing to participate and authorization could then take part in sensing tasks downloaded from private blockchain networks. This approach limits transactions to the private blockchain network to prevent privacy leakage. Because an agent provides services, participants are required to pay a service fee. Smart contracts ensure traceable, immutable, and transparent transactions record on the distribute ledger [143]. Smart contracts contribute to ensuring data quality and quantity of crowd-sensed data in blockchain-based MCS. *Workers* make decisions to engage in sensing tasks down-

loaded from either a public or private blockchain. Participants who complete sensing tasks and submit high-quality data to the MCS server would obtain rewards.

To tackle current location attacks, the authors in [259] propose a method for task distribution based on reward. This mechanism introduces tasks allocated to anonymous participants through the blockchain network to protect sensitive information. Because participants are anonymous in the MCS campaign, their identity information is protected obviously. The authors propose a blockchain-based MCS framework to address users' location privacy thread. The framework introduces an innovation task allocation mechanism that users' precise location information is not indispensable during the MCS campaign's worker selection procedure. Therefore, the current location is invisible to the MCS platform to avoid to be disclosed. The same study [259] also addresses the threat of a participant's future location. Instead of task allocation by the requester or centralized MCS platform in traditional MCS activities, several sensing tasks are provided for participants to determine which they are willing to take. Besides, a blockchain-based MCS system hides workers' identity information by default since they are anonymous. The proposed approach here comprises private blockchain networks and a public blockchain network, which is used to protect participants against re-identity attacks. A private blockchain assigns authority to particular users rather than all users, ensuring security in a private blockchain network. In a blockchain-based MCS system, participants can register themselves in different private blockchain networks that results in their location information being disclosed in particular private blockchain networks rather than a public blockchain network.

To address the location and identity privacy issue, the study in [41] proposes an approach that integrates both blockchain and smart contracts technology into an MCS platform to prevent leakage of MCS entities' (e.g., requester, participants) location and identity information. Smart contracts organize trust between different entities (e.g., requesters, participants, and an MCS platform). participants should upload their sensitive information (e.g., location and identity information) to a centralized MCS platform during participants registration and task allocation typically. In the proposed framework, smart contracts are deployed to save all interaction records in MCS activities. Moreover, transaction records do not link creators' identity information, which eradicates the possibility of user identity leakage. Smart contracts are utilized to obtain authority from the MCS platform. They prevent private information leaks from mobile users, run auctions, assign awards to participants, and charge service fees from MCS platforms.

Location privacy protection is studied in [283], which introduces CrowdBLPS, a blockchain -based MCS system, to ensure user location privacy. Based on cryptography technology in the proposed system, trustworthiness is built with the following intentions: 1) addressing intentionally altered information in the pre-registration stage, and 2) preventing location

leakage in the recruitment stage. The study in [283] presents a decentralized system model CrowdBLPS consisting of service requesters, staff, and verifiers entities. Service requesters and workers have the same responsibility as in the traditional MCS systems. A verifier participates both in verification and negotiation procedure. A verifier is a miner node selected for executing the POW process. The proposed system ensures workers' location security through a cloaked area mechanism instead of the exact location in the worker recruitment stage in the MCS campaign. Without precise location information, the cloaked area mechanism makes it difficult to determine the worker's distance and the task. The study demonstrates a method to estimate the distance between them. Given the tremendous number of potential location points uniformly distributed in the cloaked area, all points' geometric centroid is approximated to be the worker's predicted location to estimate its distance to the target. CrowdBLPS considers further insights to calculate the expected distance, which utilizes the probability that the worker could access a subarea target. Therefore, the cloaked zone becomes smaller and closer to the proper location than the initial approximation.

To address these misbehaving entities in MCS system, [104] evaluate three entities of an MCS system, i.e., requesters, participants, and a centralized MCS platform, may misbehave. Participants' misbehaviour means that recruited workers submit false sensing data or multiple solutions using different accounts for one task. As a result, these malicious users un-fairly increase their rewards. Misbehaving requesters initiate and upload a number of tasks during a short time interval to clog an MCS server providing service for other requesters. Such malicious requester affects MCS campaigns since requesters are supposed to be honest and trustworthy entities. A misbehaving MCS platform can intentionally modify the participant database to increase or decrease the reputation scores of particular participants. To do so, a misbehaving MCS platform manipulates participants' reputation scores directly, unlike indirect manipulation through injecting false readings or data poisoning to adjust participants' reputation scores over time. Thus, such incorrect and unauthenticated data impact participant recruitment in MCS. Besides, a misbehaving platform could modify assessment rules resulting in reduced award payment for workers. For instance, users deserve full awards upon the completion of a task and submission of high-quality sensing data to the MCS platform. However, a misbehaving platform modifies the evaluation rules to decline the assessment result, which leads to payment reduction in that task. The authors in [104] implements a decentralized MCS platform called SenseChain by integrating smart contracts in terms of user managed contracts (UMC), task managed contracts (TMC), and task detailed contract (TDC). All entities (e.g., requesters, workers, UMC, TMC, and TDC) are assigned a particular Ethereum address and communicate with each other. Requesters, who should register themselves in UMC, are responsible for

initiating and publishing sensing tasks. Meanwhile, the requesters communicate with the TMC by sending a task request containing the budget for the workers' payment after the task completing and a corresponding TDC for the task. Workers also need to register in UMC, who can help to inquire with TMC to obtain the consistent TDC for a specific task. Transactions during sensing activities are recorded and stored on the blockchain. UMC is responsible for storing requesters and participants' information. TMC handles tasks management. Especially, TMC is a bridging component between pending tasks and task objects. Task detail information is saved in TDC, which is transferred to TMC.

Moreover, the study in [136] proposes a blockchain-integrated quality control model for MCS. The authors present a verifier selection mechanism and a two-times consistency policy to build system credibility and integrity. On this basis, they implement a method to evaluate data quality according to the data value. The evaluation approach integrates fuzzy logic theory for remote sensing data and circuit breaking technology for protecting evaluation criteria. CrowdBC [129] is a blockchain-based decentralized crowdsourcing framework that addresses DDoS and Sybil attacks [175] due to malicious workers participating in a traditionally centralized MCS platform. CrowdBC contains the four types of actors: requester, worker, miner, and CrowdBC client. The CrowdBC client is dependent but is not dominated by third parties, like Bitcoin Core. Therefore, users can execute the CrowdBC client on their computing device. Miners provide services (e.g., adding a transaction into a block), so they charge a fee for such MCS activities. The authors then present three threat models targeting a malicious requester, a malicious worker, and a malicious miner, respectively. A false attack report stands for a malicious requester who aims to collect useful sensing data without sending awards to participants. Hitchhiking attacks refer to malicious workers who seek to be paid a reward without submitting high-quality crowd-sensed data. Finally, to obtain many more payments, malicious miners disturb routine procedures on the blockchain by branching off the chain or entering into some intrigue with requesters or workers. The authors in [18] propose a hybrid architecture that builds on blockchain solutions to identify and mitigate fake sensing activities in MCS. The platform is designed to address several types of attacks, for example, a contamination attack, which means adjusting sensors in the smart device resulting in collecting incorrect sensing data and sending false data to the MCS server. The authors in [151] proposed an anonymous blockchain-based decentralized crowdsourcing approach called ZebraLancer to address data and identity leakage during MCS activities. However, blockchain is transparent, so all nodes typically have the authority to access transactions in the chain. On the other hand, malicious users utilize the property to obtain sensitive information by observing workers' previous completed sensing tasks. Besides, malicious participants can register multiple accounts in the blockchain to continuously harvest rewards. In response,

ZebraLancer proposed an approach to an anonymous authentication and subtle linkability.

2.3 Network Intrusion Detection in IoT

IoT confronts several challenges restrain the development and application and security is one of the most critical issues [48, 271]. With more access points, attack surfaces become widespread [64], leading to more vulnerable targets against cyber-attacks even if defense mechanisms are in place [171]. There are various approaches to ensure network system security, including but not limited to firewalls, antivirus software, access control, and anti-malware [115, 233, 48, 149]. Configuration and path analysis to protect a critical infrastructure such as a smart-grid network can be named as to name one out of many examples for firewall use case [77]. In another context, the study in [7] introduces neural network-based classification using firewall logs to ensure system security.

Recently, NIDS have been widely used to detect intrusions, especially machine learning algorithms resulting in promising detection performance. ML-based intrusion detection has been investigated by various studies in IoT networks [21, 184, 186]. ML-based NIDS has been proven to enable NIDS to detect popular attacks and zero-day attacks which has also been widely investigated in the context of defensive and proactive / adversarial ML [146]. An example of a ML-driven NIDS is an XGBoost-DNN integrated NIDS [59], which has been shown to demonstrate decent performance.

The study in [214] presents a neural network-based framework to identify abnormal actions in the system so to safeguard against attacks. The authors in [145] demonstrate a NIDS framework to detect specific attacks integrating supervised (e.g., XGBoost) and unsupervised (e.g., expectation-maximization) ML algorithms whereas the authors in [185] analyze the feasibility of adopting deep learning models in NIDS. The study in [205] presents an intrusion detection system to prevent routing attacks such as sinkhole and selective forwarding attacks in IoT networks. The studies in [49, 182] demonstrate ensemble learning-based NIDS taking advantages from various individual ML models to make wise estimations. The study in [201] introduces an IDS based on the Adaboost model for DoS attack detection. The work applies a public IoT dataset MQTTset for evaluation that achieved up to 95.84% overall accuracy. Meanwhile, the authors in [236] present a framework with Adaboost for Low-Rate One-Class DoS (LRDoS) attack detection, that is more difficult to identify when compared to conventional DoS attack. Moreover, the proposed approach is adjusted according to sample weights, which tackles imbalanced dataset issues. Under NS2 simulation-based test environment, the presented method performs 97.06% detection rate for the LRDoS attack. Typically, there are various types of attack

patterns; hence, ML-integrated attack detection can be categorized into binary classification and multi-class classification. The former label all attack points as one class, whereas the latter treats each type of attack data separately. The authors in [111] propose an approach combining random tree and NBTree models for intrusion detection, which presented a better performance than individual ML algorithm under NSL-KDD. The study in [65] demonstrates a hybrid framework using Random Forest (RF) and K-means algorithms. Specifically, RF is trained via a collected dataset for misuse detection, and K-means is valid for novel attack mitigation. In [182], an adaptive approach is proposed to switch between a competent model on known attacks and another model that can detect unknown attacks better. The authors in [181] demonstrate a reinforcement learning-integrated approach for network intrusion detection in a wireless sensor network (WSN) and obtained remarkably high detection performance. Moreover, RF and an enhanced DBSCAN models are also deployed to address false negative intruder decisions in WSNs [183].

As for deep learning-integrated approaches, Researchers apply Deep AutoEncoder (DAE) in various fields such as dimension reduction, anomaly detection, and noise filtering [113]. A DAE consists of two symmetric multilayer feedforward networks (e.g., encoder and decoder). DAE is used to reproduce the input at the output layer via the encoder and decoder. The number of neurons in the output layer is the same as the number of neurons in the input layer [268]. The study in [153] demonstrates a neural network with three layers for intrusion detection that integrates DAE and some other ML algorithms. The authors utilize an improved dataset based on NSL-KDD via extracting samples in different layers (e.g., application layer, network layer, transport layer, and full layer). The highest accuracy is up to 97.83% with the proposed framework that can reduce detection load and improve detection performance. Moreover, a lightweight DAE is introduced in [124] that achieves a high accuracy up to 99% under an IoT specific dataset N-BaIoT. Fully connected neural network algorithms have proven effective at network intrusion detection with promising performance under different network types. A DNN model consists of one input layer, two hidden layers, and one output layer. The three layers are fully connected, which means neurons in each layer have connections to all active neurons in the previous layer. Assuming that there are m neurons in layer $l-1$. Fully connected neural network algorithms are extensively researched to detect network attacks [56, 67]. Specifically, the study in [30] proposes a DNN-based approach for intrusion detection in a smart home IoT network. The study in [190] demonstrates a use case on a smart factory deploying a DNN-based intrusion detection approach that is a context-sensitive based system. The authors collect data and utilize it for a ML model to train and test. The approach is applied into a real company for a test that has been proven to be a cost-effectiveness IDS. Meanwhile, the dataset is private which is difficult for other researchers to access. The authors in

[56]present a two-stage IDS BotChase targeting bot attack which integrates unsupervised ML algorithm K-means in the first stage and supervised algorithms (e.g., LR, DT, SVM, FNN) in the second stage. The Botchase can identify bots relying on various protocols, which achieves a robustness system to detect unseen attacks. However, the Botchase takes long time to converge. In [110], feed-forward deep neural networks are proposed to boost the detection accuracy of traditional ML algorithms (e.g., KNN, DT, Naive Bayes) for binary classification and multi-class scenarios.

Ensemble learning approaches have been shown to improve performance via integrating several ML techniques. Ensemble learning enables better predictive performance compared to an individual model [61, 282]. Recently, ensemble learning models attract attention to construct network intrusion system. Adaboost stands for Adaptive boosting, the first practical boosting algorithm, which is an ensemble learning used in various application to secure IoT networks [207, 234]. The study in [201] introduces an IDS based on the Adaboost model for DoS attack detection. The work applies a public IoT dataset MQTTset for evaluation that achieved up to 95.84% overall accuracy. Meanwhile, the authors in [236] present a framework with Adaboost for Low-Rate One-Class DoS (LRDoS) attack detection, that is more difficult to identify when compared to conventional DoS attack. Moreover, the proposed approach is adjusted according to sample weights, which tackles imbalanced dataset issues. Under NS2 simulation-based test environment, the presented method performs 97.06% detection rate for the LRDoS attack. RF is an ensemble ML algorithm that consists of a number of decision trees [198]. Each decision tree gives a prediction result and an overall result via voting on by all decision trees. For instance, it is assumed that three decision trees are applied in RF for binary classification, class A and class B. Two of the trees give prediction results at a point belonging to class A, while one of three trees estimates the sample belongs in class B. Then RF considers the data as part of class A since the majority of members voted for class A. Multiple class classification follows in the same manner as binary classification. The RF algorithm is deployed to detect network attacks in Ethernet IoT and self-designed networks [88, 234]. The study in [16] demonstrates a two-stage detection framework for blackhole and DoS attacks in mobile IoT systems. The authors collect data in a designed network and feed the data into RF and other ML models. The numeric results show an F1 score in the range of 93% to 99.36% in identifying the two attacks. In [176], IoT botnet attack mitigation is investigated integrating feature selection and post-hoc local analysis stages in RF and some other ML algorithms working procedures. XGBoost is one of the most popular ensemble ML algorithms, which is based on decision trees that uses a gradient enhancement framework [45]. The study [234] report on an XGBoost-based classification method to mitigate network attacks with a detection performance of 97%. The authors combine XGBoost and LSTM

together to construct a novel framework in order to ensure IoT devices security [244]. It is worth noting that a real IoT system is established for performance evaluation. The proposed approach achieves promising performance for vulnerability exploiting, malware infection, abnormal operation, and memory leak. Moreover, the paper in [75] presents a vote-integrated ensemble learning approach that combines on base ML models such as DT, RF, KNN, and DNN to boost performance.

Ensemble learning can be implemented in either homogeneous or heterogeneous manner [210]. A homogeneous ensemble is composed of a single type of base classifiers that employs the same feature selection technique on a variety of training samples and allocates the dataset across multiple nodes [227]. RF is a type of homogeneous ensemble learning which contains several decision trees as base-classifiers. Meanwhile, Adaboost and XGBoost are kinds of homogeneous ensemble learning which contains homogeneous weaker learners. On the other hand, a heterogeneous ensemble contains different classification algorithms and uses the same training dataset for base-classifiers [15]. A heterogeneous ensemble has been reported to be robust, precise, and generalized techniques to solve pattern recognition problems [230]. The authors in [211] demonstrate that different base classifiers contribute diverse classification hypotheses in feature space which is crucial to building a reliable, robust, and generalized ensemble method. Therefore, we focus on heterogeneous ensemble models in this thesis to address multi-class attack detection problems so to obtain a higher detection accuracy or lower prediction costs. Ensemble learning provides an opportunity to tackle the one-fits-all-sizes issue by integrating several MLs and taking advantage of forming different MLs. The study in [62] demonstrates an intrusion detection model based on CNN in comparison to the traditional ML models (e.g., RF, SVM, DBN, and LSTM). The authors illustrate numerical results using multi-class classification to detect attacks in the NSL-KDD dataset. The implemented ML models reveal diverse false positive rates (FPR) and detection rates (DR) even under the same dataset.

Related work discusses empowering host-based intrusion detection (HIDS) to detect particular attacks that are challenging for an NIDS. Moreover, HIDS eliminates monitoring of the entire network traffic by leveraging host resource analysis, which makes HIDS appealing to researchers. For instance, the study in [76] proposes a host-based system to detect network intrusion, which integrates user space and kernel space data in intelligent equipment and extracts data into numerical arrays for training ML models. Moreover, in [87], the authors investigate observing information from system logs which are host-based features, and show that such systems could classify suspicious activities with high accuracy. Furthermore, the authors in [202] utilize host-based information to develop a detection system via organizing statistical data and ML models. However, hybrid network-based and host-based intrusion methods are limited, such as the study in [148].

2.4 Conclusion

ML-based approaches have been previously introduced to detect fake tasks in MCS systems for enhanced security. However, achieving high performance in fake task detection is challenging due to specific vulnerabilities like limited dataset samples, insufficient features, and concerns with centralized MCS systems. Additionally, the issues related to centralized architecture in MCS are crucial. The advent of blockchain technology offers a decentralized solution, making it worthwhile to explore its integration into MCS systems for decentralized functionality. Moreover, intrusion detection has emerged as a significant concern in IoT systems. Identifying network intrusions is vital to prevent attacks and ensure the overall security of IoT systems. However, there is limited research on leveraging ML algorithms to make informed decisions for improved detection performance. Furthermore, there is a lack of literature on hybrid network and host data for intrusion detection in IoT networks, making it important to focus on these two issues. In this thesis, the aforementioned issues will be thoroughly examined and addressed through proposed approaches.

Chapter 3

Machine Learning-based Approaches to Secure MCS In IoT

MCS platforms are vulnerable to various attacks as they build on non-dedicated and ubiquitous properties. Especially, submission of fake tasks with the aim of clogging participants device resources as well as MCS servers is a crucial threat to MCS platforms. An MCS model has two participant modes: requesters and workers/devices [246]. Typically, an MCS campaign starts with a requester prepare tasks and send them to the MCS platform. A requester normally uses smart devices to create MCS sensing tasks. As introduced in [39], MCS system typically can be designed with layered architecture, including sensing layer, communication layer, data layer, and application layer. Participants use smart devices with various sensors to collect sensing data in sensing layer. The collected sensing data is transferred via participants using mobile devices in communication layer. Mobile devices commonly come with multiple radio interfaces, such as cellular, WiFi, and Bluetooth to transfer data. After that, data layer is responsible for storage, analysis, and processing, that includes collected sensing data, sensing task information and other information. Application layer includes considerations of task-related and user-related factors to plan and structure a MCS campaign. As illustrated in Fig. 3.1, an MCS campaign works as following steps: (1) requesters prepare tasks and send them to the MCS platform; (2) MCS platform collects tasks from requesters; (3) platform recruits users and distribute tasks to participants; (4) data is collected and uploaded by recruited participants; (5) MCS platform assembles uploaded sensing data, analyze data, and process data; (6) MCS platform generates a report for requester. However, malicious attackers publish illegitimate tasks to MCS platform so MCS campaign can not work smoothly as expected steps. Fig. 3.1 demonstrates attackers sending fake tasks to MCS server with the motivation of draining

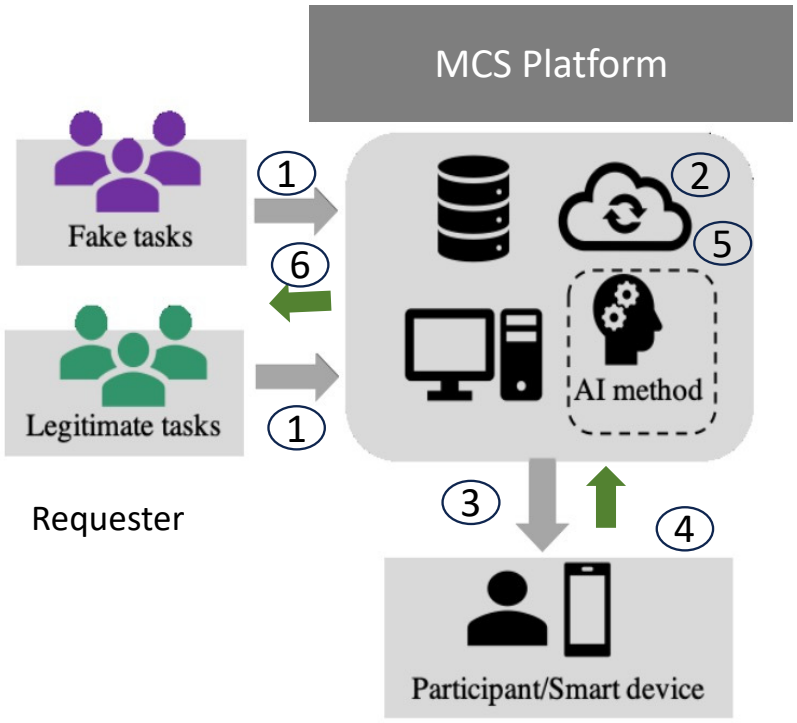


Figure 3.1: Fake task attack on an MCS server.

more energy from users' devices and clogging MCS server. Section 3.1 demonstrates tasks characteristics comprehensively, including fake tasks and legitimate tasks. ML-based approaches are widely investigated to build attack detection systems for fake tasks detection and ensure MCS systems security. However, obtaining a promising fake task detection performance is challenging. Specifically, in order to hide fake tasks and avoid detecting, cyber criminals typically inject limited attack samples (e.g., fake tasks). Typically, a limited sample of a specific attack can be problematic because it may not represent the full range of behaviors, variations, or consequences associated with that attack. Meanwhile,AI algorithms encounter difficulties when attempting to learn from small datasets due to a variety of factors, including overfitting, noise, outliers, and sampling bias. These challenges can lead to a model that lacks effectiveness [240].

In order to identify fake tasks in MCS activities, an intelligent attack scenario is designed in Section 3.2 based on the Self-Organizing Feature Map-based fake task attacks where the task submissions are scarce, which is a degrading factor for the performance of an analytical platform, alongside the roadblock of class imbalance. To cope with the

imbalance between illegitimate and legitimate tasks, repetitive oversampling is applied on scarce data of sensing task submissions to a MCS platform (generated via CrowdSensim using realistic physical features of a small city in Canada), implement a Principal Component Analysis (PCA) module, and model a Deep Belief Network (DBN) with various design parameters to filter our fake task requests submitted to the MCS platform. Meanwhile, imbalanced dataset is common in real world problems such as failure diagnosis, spam detection, anomaly detection, fraud detection and medical diagnosis, which affects the quality of deep learning and machine learning. Therefore, to tackle imbalanced dataset problem, authors in [264] propose DBN based re-sampling support vector machine ensemble learning paradigm to solve the imbalanced data problem. As well as cost-sensitive Deep Belief Networks is demonstrated for imbalanced classification problems [266].

Meanwhile, the second fake task detection method is proposed in Section 3.3 that utilises K-means-based feature extraction integrating ensemble and deep learning algorithms. ML-based method to tackle malicious task attacks in MCS. The proposed system can predict the legitimacy of a task in order to avoid fake tasks consuming battery power from users' devices. Fake task attack is designed focusing on specific locations where illegitimate tasks inject into the MCS system. To address this issue, two artificial intelligence (AI)-driven algorithms are deployed: a Deep Belief Network (DBN) and an ensemble learning method (i.e., Bagging). Moreover, k-means-based unsupervised classification is utilised to provide region-specific inputs to the classification models. With the extracted feature, it is helpful to train DBN and Bagging better and make more precise prediction for fake tasks.

Lastly, a novel approach, based on horizontal Federated Learning (FL), is discussed in Section 3.4 to identify fake tasks that contain a number of independent detection devices and an aggregation entity. Detection devices are deployed to operate in parallel with each device equipped with a ML module, and an associated training dataset. Furthermore, the aggregation module collects the prediction results from individual devices and determines the final decision with the objective of minimizing the prediction loss. Loss measurement considers the lost task values with respect to misclassification, where the final decision utilizes a risk-aware approach where the risk is formulated as a function of the utility loss.

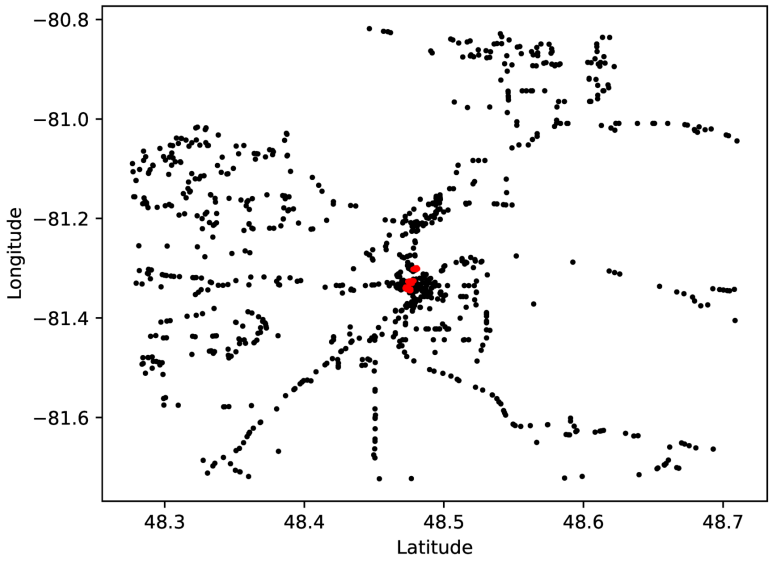
3.1 Characteristics of Sensing Tasks Submitted to the MCS platform

3.1.1 Sensing Task Introduction

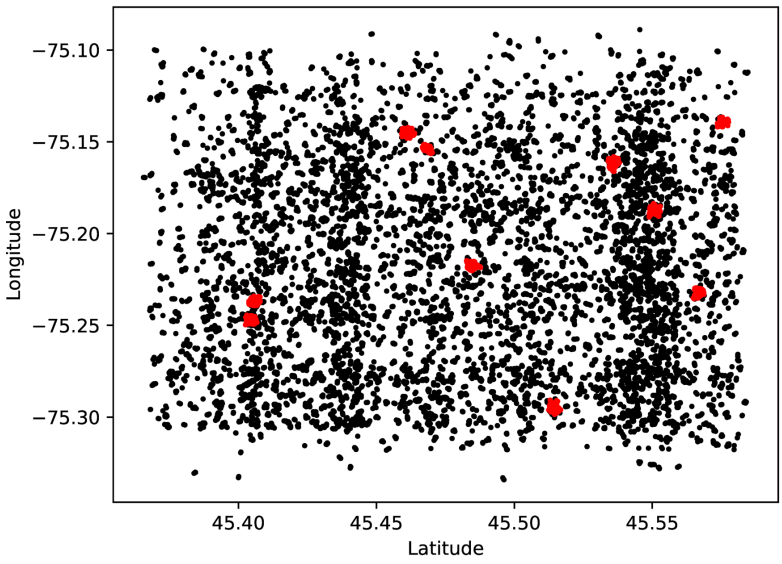
A MCS task is a specific request or job assigned to participants in a crowdsensing network. These tasks typically involve collecting data or performing actions using mobile devices, such as smartphones or wearables, in a distributed and collaborative manner [173, 241]. A MCS task introduces particular requirements to MCS platform, such as data quality, task duration, incentive/rewards, task distribution, task evaluation and so on. There two kinds of tasks, including legitimate task and fake task sent to MCS platform, as shown in Fig. 3.1. Legitimate tasks are submitted by normal requesters who aim to collect specific information in particular location. However, fake task objective is not to assemble data using MCS system, that aims to consume more resources from participants' equipment and clog the MCS sensing server to prevent the MCS server from providing service to normal users. Fake task are described firstly in [272], which is a critical threat that impacts an MCS platform and sensing data providers. To prohibit legitimate tasks from being dominated, the adversary intends to inundate the targets with extraneous duties, also known as fake tasks. A crucial assumption for fake task definition is that fake tasks are more likely to begin during peak hours of the day. The definition of peak hour here follows that of the utilities. If the DoS assault persists for an extended period of time in the MCS system, the platform will begin to lose users' desire to engage in MCS campaigns. In order to preserve the integrity of the MCS system, fake tasks must be anticipated before to being given to any participant. It is logical to presume that all fake tasks originate in particular areas of the city. Henceforth, we will refer to them as the attack zones or attack regions. Regarding legitimate tasks, geographical location are chosen at random; consequently, legitimate tasks may be distributed in attack zone which will be illustrated in Fig. 3.2.

The objective of fake tasks is two-fold: 1) draining batteries of user devices and clogging their sensing, memory and processing resources 2) clogging the sensing servers in the MCS platform. Therefore, fake tasks are important threats to an MCS platform and user devices. To achieve these aims with maximum possible impact, the features of fake tasks have been designed to cover longer duration, aggregation during the peak hours when communication channels are utilized more as suggested in [272]. In addition, a Self Organizing Feature Map (SOFM) is applied to model a smarter fake task injection in the recent work by Zhang et al. [275]. six coordinates of the aggregated population in a city are extracted according to users' routine data of six days.

As introduced in [272], a task contains several features {'ID', 'latitude', 'longitude', 'day', 'hour', 'minute', 'duration', 'remaining time', 'battery requirement %', 'Coverage', 'legitimacy', 'GridNumber', 'OnpeakHour'}. In this thesis, several machine learning or deep learning algorithms are trained by using these features. 'ID' is a unique number for a task. The combination of 'latitude' and 'longitude' indicates task distribution position. 'GridNumber' is extracted via sensing area map splitting, which also stands for position information. 'GridNumber' has been introduced to provide a quantitative representation of coordinates. This is achieved by dividing a map into small rectangular partitions, based on the minimum and maximum latitude as well as the minimum and maximum longitude of the city map. Each of these partitions forms a grid, with an approximate length of around 600 meters [46]. Feature 'day', 'hour,' and 'minute' describe a time of task distribution. Meanwhile, 'OnPeakHour' stands a task running during peak hours or not. The 'OnPeakHour' attribute stands for if a task is distributed from 7 am to 11 am, defined as busy communication time. 'Duration' shows the time required for completing a task. 'Remaining time' represents the rest time to complete a sensing task. 'Battery requirement' outlines the consumption of batteries in intelligent devices to finish a task. 'Coverage' means a task advertising distance from a published location. A task is labeled as either legitimate or fake to feature 'legitimacy'. Fake task and legitimate task consist of various properties for these features. The parameters as shown in Table 3.1 are built on fake tasks and legitimate tasks properties and rationality analysis. The parameters in Table 3.1 demonstrate the core difference between fake tasks and legitimate tasks. Specifically, fake tasks are generally distributed during peak hours (e.g., 7 am to 11 am) to allure more victims. Thus, 'Hour' parameter is configured as 80% fake tasks distributed over the time interval from 7 am to 11 am. On the other hand, the distribution hour of most legitimate tasks (92%) is randomly distributed in the daytime from 6 am to 11 pm since most normal service requesters submit legitimate tasks in their daily working time. Meanwhile, fake tasks aim to drain more energy, so fake tasks last a longer duration and higher percentage of battery consumption. Therefore, 'Duration' attribute of fake tasks is set up as 70% by taking 40 to 60 minutes and 30% lasting 10 to 30 minutes while the duration time of legitimate ones is uniformly distributed in 10 to 60 minutes. 'Battery' parameter follows a similar rule as 'Duration' for fake and legitimate tasks. The CrowdSenSim tool [72] is utilized to simulate MCS activities and generate sensing tasks, including fake tasks and legitimate tasks. The CrowdSenSim tool allows a maximum six-day simulation, so 'Day' attribute is among 1 to 6.



(a) Timmins dataset



(b) CR dataet

Figure 3.2: Task distribution regarding longitude and latitude features in Timmins dataset and CR dataset. Black dots denote legitimate tasks and red dots represent fake tasks.

Table 3.1: Class-specific simulation settings

	Illegitimate Tasks	Legitimate Tasks
Day	Uniformly distributed in [1, 6]	
Hour (task)	80% : 7am to 11am; 20%: 12pm to 5pm	8%: 0pm to 5am; 92%: 6am to 23pm
Duration (task)	70% in {40, 50, 60} [minutes]; 30% in {10, 20, 30} [minutes]	Uniformly distributed over {10, 20, 30, 40, 50, 60} mins
Battery usage of smartphone for a whole task	80% in {7%-10%}; 20% in {1% -6%}	Uniformly distributed in {1%-10%}

3.1.2 Dataset Generation

The class-specific simulation settings is adopted from [274] as shown in Table 3.1 and use the coordinates obtained from SOFM modeling as the centers of fake task submission zones (also called attack areas/regions). The dataset with sensing tasks is created via the CrowdSenSim tool, and simulation configuration is introduced in Table 3.1. According to task characteristics, the CrowdSenSim tool generates tasks by using Timmins and Clearance-Rockland (CR) geography to form dataset in this chapter. Timmins is a small town in Canada, while CR represents a big city in Canada. The two datasets are imbalanced, with the majority of legitimate tasks and minority of fake tasks. In order to demonstrate the difference of the two kinds of task, Fig. 3.2 shows legitimate tasks and fake tasks distribution regarding geographical location (e.g., longitude and latitude features) under the Timmins dataset and CR dataset. In the Fig. 3.2, on the one hand, legitimate tasks are represented by black dots, which show a random distribution characteristic; on the other hand, there are 10 attack zones for fake tasks distribution as denoted via red dots Fig. 3.2. Specifically, in the Timmins dataset, there are three attack zones and zones are close to each other, While in the CR dataset, there are multiple attack zones far away from each other. Attack zone is one of the most significant properties and assumptions for fake tasks. Based on task definition and class-specific simulation setting in 3.1, CrowdSenSim simulation tool is applied to generate $1,000$ tasks in Timmins, with 903 legitimate tasks and 97 fake tasks. In CR, total $14,484$ tasks are generated by Crowdsensim tool with $12,587$ legitimate tasks and $1,897$ fake tasks respectively, which is a typically imbalanced dataset. Table 3.2 presents the two datasets fake task and legitimate task distribution. In this thesis, only CR and Timmis datasets are used in MCS fake task detection. One of

Table 3.2: Timmins and CR dataset distribution

	Fake	Legitmate
Timmins	97	1897
CR	903	12587
Total	**1000**	**14484**

our future works is to apply additional datasets to verify generalization of the following proposed approaches.

3.2 Deep Belief Network-based Fake Task Mitigation for MCS under Data Scarcity

In order to draw the motivation for this study, Fig 3.3 illustrates the distribution of illegitimate and legitimate sensing tasks on an 3-dimensional space whereas Fig. 3.4 illustrates the prediction of legitimate and illegitimate tasks when the proposed PCA-preceded and DBN-based framework for the detection of fake sensing tasks has been applied. DBN is selected since it demonstrates advantages through a two-phase training process involving pre-training and fine-tuning. During the unsupervised pre-training phase, each layer is trained in a greedy, layer-wise manner, which frequently results in improved convergence and generalization during the subsequent supervised fine-tuning phase. This is especially beneficial when dealing with limited datasets. Additionally, the DBN model mitigates the risk of overfitting by leveraging a small amount of labeled data during the fine-tuning process[91]. The DBN model is deployed to detect illegitimate tasks from legitimate tasks which can be considered as binary classification problem. Currently, DBN is widely used in image and audio classification [2] [3] and exhibits promising. Originally proposed by G. Hinton [90, 79], DBN can be considered as stacking version of multiple Restricted Boltzman Machines (RBMs).

3.2.1 Deep Learning-based Detection of Fake Sensing Tasks

The focuses on the DBN model to detect malicious tasks and Fig. 3.5 represents its building blocks. DBN were created as a response to the issues experienced while training classic neural networks in deep layered networks, such as training slowly and demanding a large number of samples to train neural networks. DBN is the result of a trade-off between the

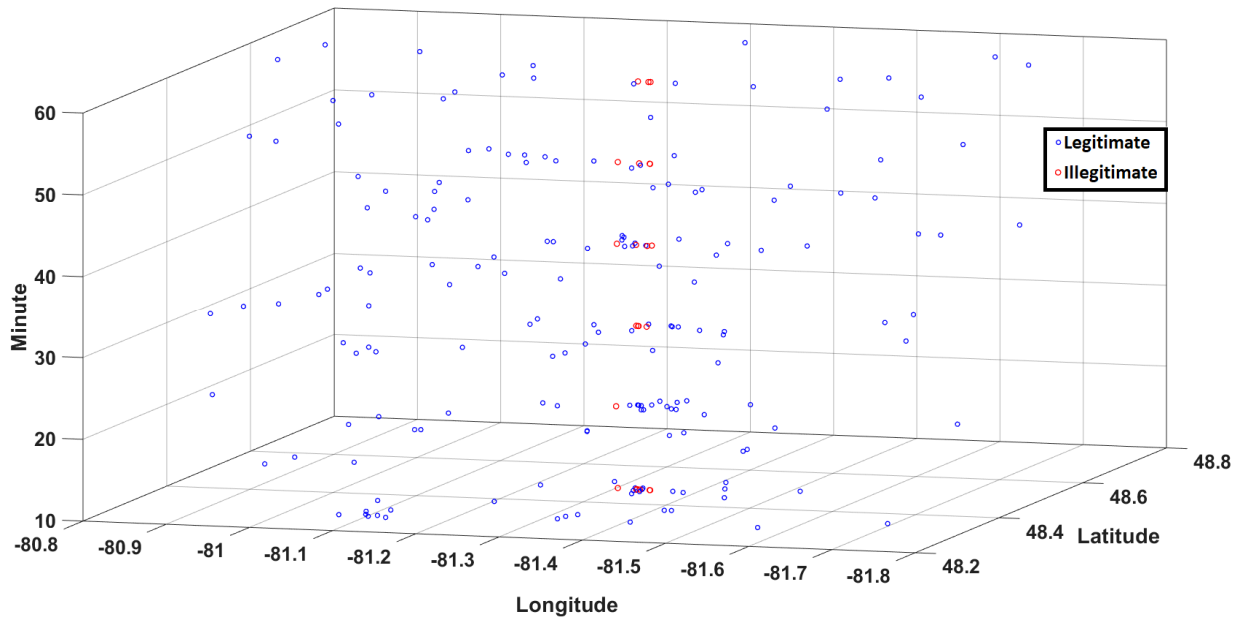


Figure 3.3: Legitimate and illegitimate sensing tasks in terms of latitude, longitude and minute features in the test set.

modelling capacity and computational complexity of ML models for graphic application scenarios, and many modern AI algorithm are the neural equivalents of historical linear models [239]. DBN is robust in classification and productive in its use of hidden layers when compared with Multilayer Perceptron (MLP) [105]. Moreover, DBN is proven to boost performance in handwriting classification when compared with feedforward neural networks [92]. Furthermore, DBN is easy to be applied in to new problem in the future so DNB is valuable to be investigated in MCS systems. System architecture in the detection of illegitimate tasks is adopted from [274] where the authors presented integration of machine learning with MCS to recognize illegitimate tasks. The MCS simulator generates original tasks with m features. Following upon the generation of m features, PCA analyzes original tasks and reduces dimensions to n features. The n feature dataset is split into two parts: training and test datasets. The following step is to over-sample the training dataset, and then apply it to the DBN model for training. The next step is to test the sensing task data on the DBN model to predict whether a sensing task is legitimate or not. If the DBN prediction result is lower than expected, the DBN parameters could be updated and adjusted.

It is worth to note that this work investigates performance improvement of DBN-

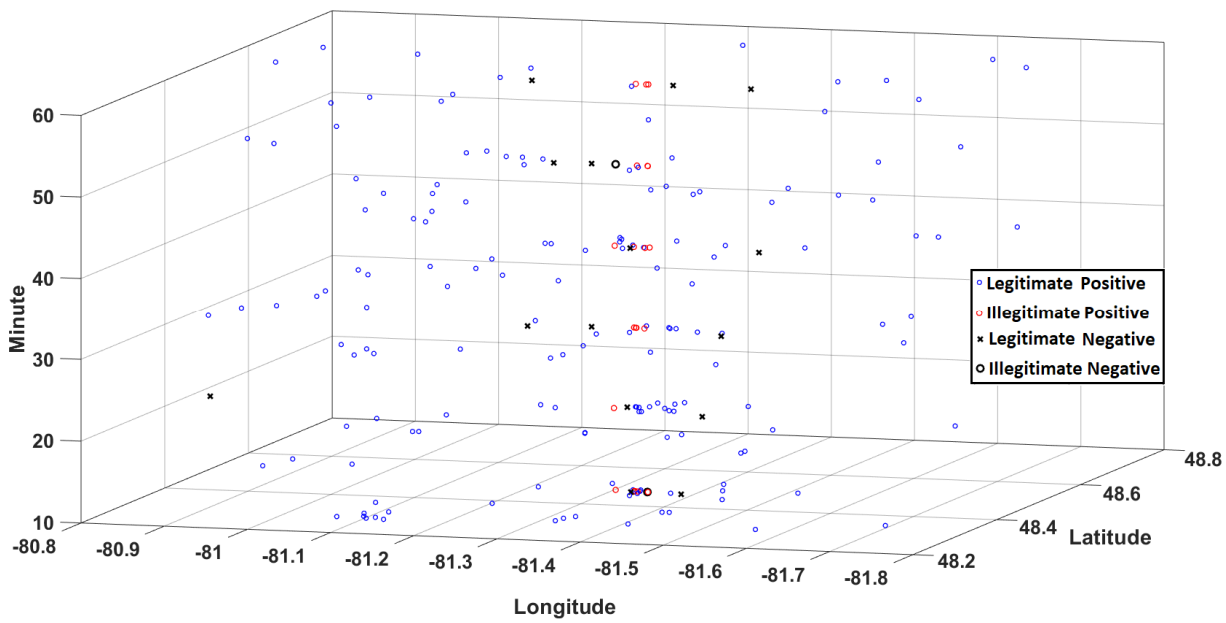


Figure 3.4: Legitimate and illegitimate sensing tasks prediction results after applying PCA and DBN in terms of latitude, longitude and minute features.

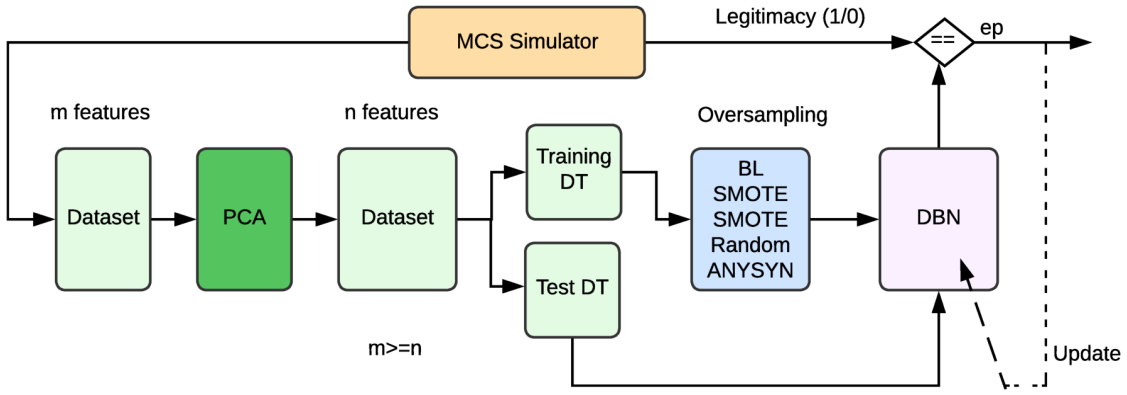


Figure 3.5: DBN based system overview

based fake sensing task detection under the scarcity of the data regarding sensing task submissions. Malicious requesters seek to conceal their identity, which often results in a restriction on the number of fake tasks they submit to the MCS platform. This restriction, in turn, causes a shortage of fake tasks. Therefore, the dataset generated by the MCS simulator has only 1000 tasks, which is a low volume dataset for a deep learning network especially when compared to [218] where dataset with $14,484$ tasks is applied as described in Table 3.2. Another challenge is the data imbalance problem as legitimate tasks form an overwhelming portion of the entire sensing task submissions, i.e. the proportion of legitimate to illegitimate tasks is 89%:11%. To cope with this, over-sampling methods are applied to fix the imbalance problem. Meanwhile, PCA is used to analyze the feature set and reduce dimensions to speed up the learning process significantly [170]. Furthermore, it is common to see that datasets contain noise or irrelevant information in the form of features with low variance. PCA identifies these low-variance directions as less informative and thus reduces their impact on the model. This can lead to a cleaner and more robust representation of the data [155]. Moreover, fewer features mean simpler models. Simplified models are less prone to overfitting, as there are fewer parameters to estimate. This helps models generalize better to unseen data. Thus, PCA is deployed for MCS dataset in this thesis to eliminate irrelevant information, address the overfitting issue and speed up the training procedure.

Principal Component Analysis (PCA)

The rationale behind integrating PCA is to reduce the number of inputs to the ML model in the dataset so to speed up the training process on the deep network [170]. Principal Component Analysis (PCA) was invented by Karl Pearson in 1901 which is a linear dimensional reduction. It is widely used and becomes popular in the area of extracting features which reduces high-dimensional inputs to low-dimensional ones. It claims one of the best methods in regard to dimension reduction [170]. PCA assists in identifying a smaller amount of characteristics that describe original datasets in a condensed manner, preserving up to a specific proportion of its variance based on the number of new features. This transformation is specified such that the first principal component has the most possible variance and each subsequent component has the greatest possible variance.

Traditional PCA is linear dimensionality reduction using Singular Value Decomposition (SVD) to reduce the data's dimensionality and project it into a lower dimensional environment. Before performing the SVD, the input data is centred but not scaled for each feature. There are variable PCA algorithms such as Kernel Principal component analysis (KPCA) [208], Sparse Principal Components Analysis (SparsePCA) [103], and Incremental

principal components analysis (IPCA) [203]. Specifically, KPCA utilizes kernels to accomplish non-linear dimensionality reduction which is different than classical PCA. Moreover, SparsePCA enhances the traditional approach of PCA for dimensionality reduction by including sparsity patterns into the input variables. The fact that conventional PCA are often linear composites of all input variables is a major drawback. This issue is solved by sparse PCA, which finds linear combinations including only a few input variables. If a dataset to be decomposed is too vast to store in memory, IPCA is generally utilised in place of PCA. IPCA creates a low-rank estimate for the input data using a memory size that is independent of the number of samples in the input data. Memory utilisation may be controlled by adjusting the batch size, which is based on the input data characteristics.

The conventional PCA is employed in this work to analyze the sensing task data to generate new dimensional dataset which first Principal component contains the largest possible variance and the variance of the following components degrades gradually. In this work, the key advantage of PCA is to increase the efficiency of the DBN model through smaller-dimensional inputs [206]. More specifically, a larger learning rate and a smaller number of epochs for the modeled DBN can be used with the help of PCA to degrade the time for model training eventually. As reported in the next section, the PCA is applied to analyze tasks with 6, 7 and 11 input features so to reduce to the most suitable dimensions respectively. Therefore, a performance test is presented with the new dimensions of the DBN model in Section 3.2.2.

Oversampling Methods

In the considered scenario, besides the scarcity of the data regarding sensing task submissions, the scarce data is also imbalanced. There are two directions to balance the dataset: enlarge the minority dataset through over-sampling; reduce the majority data through under-sampling. Since the condition of scarcity of sensing task submissions is considered in the MCS platform, it is viable to proceed with oversampling. When dealing with oversampling techniques in the context of imbalanced datasets, it is crucial to carefully consider the boundaries or limits of oversampling to strike a balance between addressing class imbalance and avoiding potential issues [238]. In this thesis, the oversampling boundary specifies equal composition of minority and majority classes. Specifically, oversampling techniques aim to enlarge the minority class samples until it reaches the same size of the majority class. The Synthetic Minority Over-sampling Technique (SMOTE) [152], Borderline SMOTE [84], Random over-sampling [164] and Adaptive Synthetic over-sampling (ADASYN) [83] are the widely known and common over-sampling methods , which are adopted and tested in our proposal. All of these oversampling methods differ from each

other with their insertion methods. The SMOTE aims to find the nearest neighbors for each minority class data followed by the generation of random points to the nearest neighbors of the sample. ADASYN can be considered a similar scheme to SMOTE with minor improvements as it adds a small random value to the data points. Borderline SMOTE is a variant of the SMOTE algorithm where a minority class borderline aims to be detected and considered to generate new synthetic samples. Lastly, the Random over-sampling method enlarges the minority class by picking samples of minority classes and reusing them multiple times. It is indeed essential to choose the most suitable over-sampling method for the highest possible performance of the DBN. To this end, Section 3.2.2 presents the performance comparison under DBN with different over-sampling methods.

3.2.2 Performance Evaluation

DBN model settings under the MCS Dataset

In this evaluation section, the DBN model on the MCS dataset with 1000 tasks and 80% of them are used for training process and 20% of them are used for the test to demonstrate the ML performance. The main parameters that affect the performance of DBN in the detection of fake sensing tasks are the number of inputs, hidden layers and neurons. More specifically, 2, 4 and 6 hidden layers with 32, 64 and 128 neurons are evaluated respectively. To simplify the hidden layer structure, all layers contain the same number of neurons to satisfy similar conditions for performance evaluation. For example, if 32 neurons are used in 2 hidden layers for a test, then each layer contains 32 neurons. Since the DBN model consists of multiple RBMs as stated earlier, RBM parameters are the key factors in order to improve the prediction performance of the DBN. Without PCA, setting the learning rate and number of epochs at 0.0006 and 40, respectively, can make a compromise between the training time and prediction performance. On the other hand, if PCA is applied, the RBM learning rate is set at 0.01 and the number of epochs is set at 20. With the help of PCA, stability of each run is increased so the use of the larger learning rate and smaller epochs can be possible during the training. Thus, PCA is applied in this thesis to reduce training time by using a large learning rate.

Fake sensing task detection by DBN with Different Over-sampling methods

In this section, DBN accuracy in the detection of fake sensing tasks is analyzed by applying Borderline SMOTE, SMOTE, ADASYN and Random over-sampling methods respectively. The DBN is designed with two hidden layers, 32 neurons in each and other model

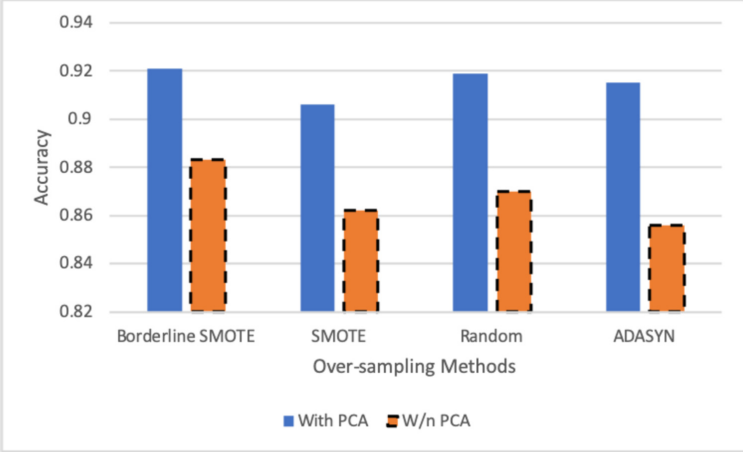


Figure 3.6: Accuracy comparison of different over-sampling methods with and without (w/n) PCA

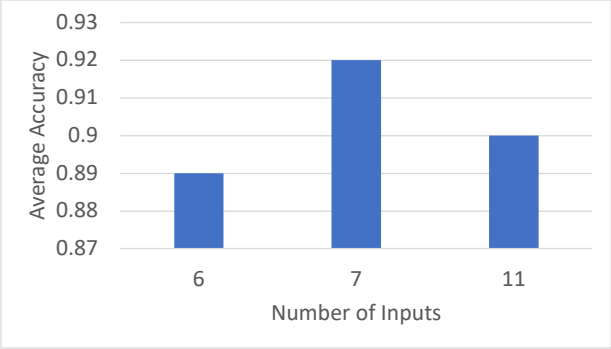
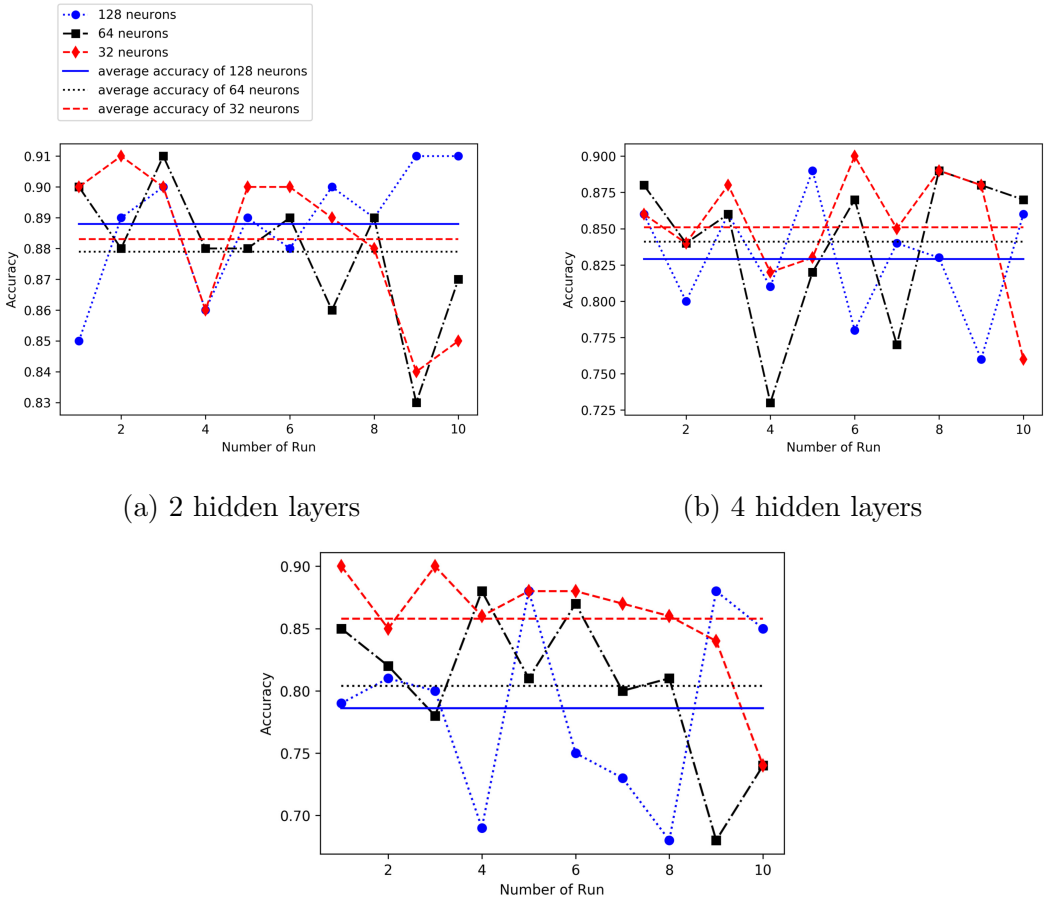


Figure 3.7: 10 rounds average accuracy comparison for 6, 7 and 11 (original) inputs to DBN preceded by PCA using 2 hidden layers with 32 neurons.

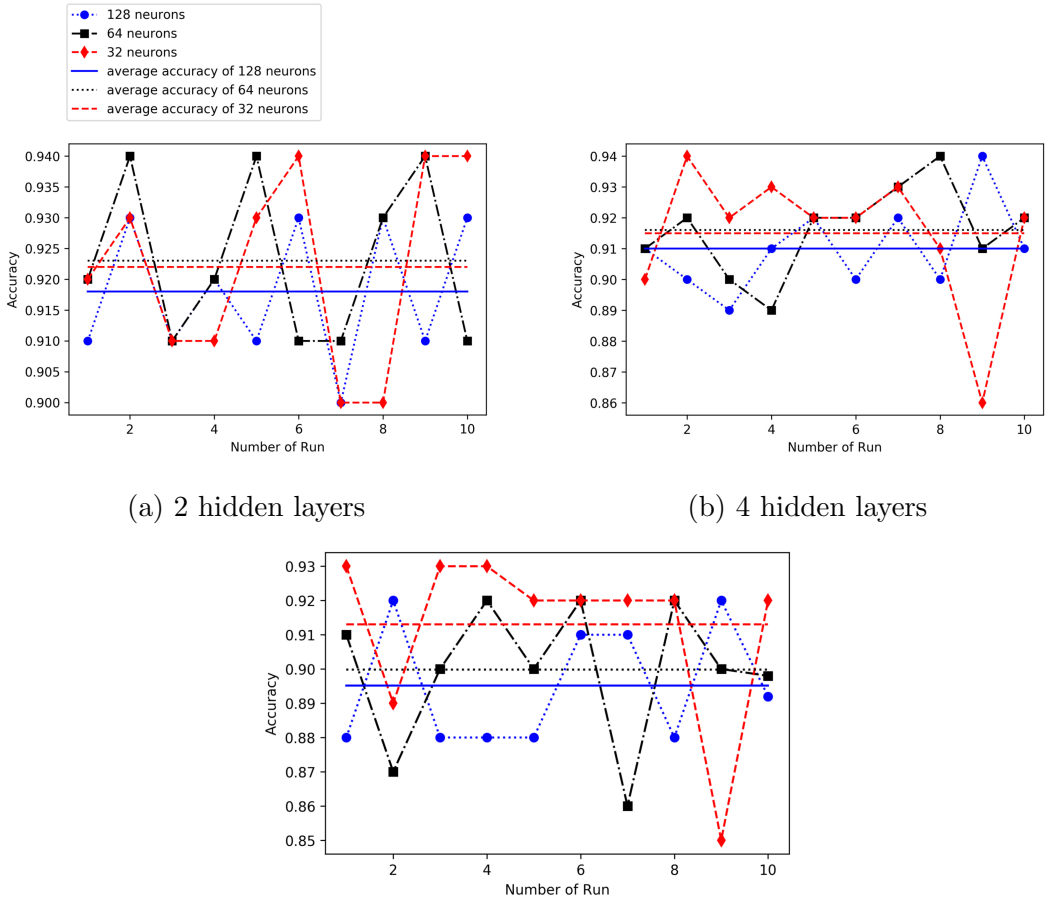


(a) 2 hidden layers

(b) 4 hidden layers

(c) 6 hidden layers

Figure 3.8: Accuracy comparison of hidden layers 2, 4 and 6 and number of neurons 32, 64 and 128 in DBN without PCA



(a) 2 hidden layers

(b) 4 hidden layers

(c) 6 hidden layers

Figure 3.9: Accuracy comparison of hidden layers 2, 4 and 6 and number of neurons 32, 64 and 128 in DBN preceded by PCA

Table 3.3: PCA analysis (variance) results for various inputs. LAT:Latitude, LOT:Longitude, H:Hour, DUR:Duration, OPH:OnPeakHours RES:Resources, RTime:RemainingTime, GNum:GridNumber, COV:Coverage

Input	LAT	LOT	H	DUR	OPH	GNum	RES	Day	Min	RTime	COV
6	0.526	0.213	0.135	0.127	0	0	N/A	N/A	N/A	N/A	N/A
7	0.431	0.182	0.173	0.11	0.103	0	0	N/A	N/A	N/A	N/A
11	0.27	0.217	0.151	0.12	0.11	0.069	0.064	0	0	0	0

Table 3.4: Overall results of 10 runs and structural details with and without PCA. The structure column indicates the number of hidden layers and the number of neurons in the each hidden layer. Most important 7 inputs are used for without PCA and after applying PCA, these 7 inputs are reduced to 5.

Structure	Pre	RC	F1	Acc	Pre	RC	F1	Acc
With PCA or Not	No	No	No	No	Yes	Yes	Yes	Yes
32-32	0.93	0.881	0.881	0.881	0.943	0.921	0.927	0.921
64-64	0.929	0.879	0.893	0.879	0.942	0.923	0.928	0.923
128-128	0.931	0.888	0.90	0.899	0.936	0.918	0.922	0.918
32-32-32-32	0.92	0.851	0.872	0.851	0.943	0.915	0.921	0.915
64-64-64-64	0.913	0.840	0.82	0.841	0.933	0.916	0.919	0.916
128-128-128-128	0.907	0.829	0.851	0.829	0.931	0.91	0.914	0.91
32-32-32-32-32-32	0.929	0.858	0.879	0.858	0.941	0.913	0.922	0.913
64-64-64-64-64-64	0.92	0.805	0.837	0.804	0.933	0.898	0.908	0.898
128-128-128-128-128-128	0.919	0.785	0.822	0.786	0.932	0.892	0.901	0.892

parameters are set up as Section 4.2.2 introduced. As seen in Fig. 3.6, the Borderline SMOTE outperforms the other oversampling methods whereas SMOTE demonstrates the worst performance. ADASYN and Borderline SMOTE are variants of SMOTE with minor improvements. Therefore, ADASYN and Borderline SMOTE improves the detection accuracy under SMOTE. On the other hand, DBN is not preceded by PCA in the detection of fake sensing tasks, as shown in Fig. 3.6, Borderline SMOTE demonstrates the best accuracy. However, the performance of ADASYN and SMOTE are almost similar, which is different from applying PCA. In any case, the detection of fake sensing tasks in the MCS platform by DBN can be improved with the PCA-integrated design. The study in [84] presents the details of Borderline SMOTE algorithm and shows Borderline SMOTE

out perform SMOTE and random over-sampling methods [84]. That is the reason that Borderline SMOTE outweighs others in Fig. 3.6.

PCA Analysis Performance

In this section, PCA is integrated to analyze input features and determine the most suitable features after applying dimension reduction. More specifically, PCA analysis is applied to 6, 7 and 11 features respectively. The first row in Table 3.3 implies that 4 dimensions are feasible within small amount of data loss. The 7 and 11 inputs analyses follow the same guidelines with PCA to reduce the number of dimensions to 5 and 7, respectively, referring to the second and third rows of Table 3.3. PCA captures the variance in the data, and this variance is a key factor in determining the importance of each component. For instance, if original dataset has six inputs, the importance of the last two components is almost 0 in 3.3. Hence, PCA transforms a six-input dataset into a four-input dataset. Simultaneously, it converts a seven-input dataset into a five-input dataset and an eleven-input dataset into a seven-input dataset. Fig.3.7 illustrates the performance of 6, 7 and 11 original inputs reducing to 4, 5 and 7 dimensions respectively in 2 hidden layers with 32 neurons. The dataset with 7 original inputs outperforms the others. Therefore, the conclusion is that the 7 inputs (Latitude, Longitude, Hour, Duration, OnPeakHour, GridNumber and Resouces) represent comprehensive characteristics of sensing task submissions, and keep tight correlations between the inputs and outputs. Although the MCS dataset is not complex with limited number of features such as 6, 7, or 11, PCA raises fake task detection performance as shown in Table 3.4. The reason is PCA remove some noise information and keep the majority information in the original dataset. It is helpful for ML algorithms to prediction so a better performance achieves.

The impact of PCA module on the detection of fake sensing tasks

Results in this section are based on the 7-input dataset without PCA, and 5 reduced inputs when PCA precedes the DBN. It is worth to note that Borderline SMOTE over-sampling method is used to cope with the imbalance between legitimate and fake sensing tasks submitted to the MCS platform. Fig.3.8 illustrates accuracy comparison without applying PCA for 2, 4 and 6 hidden layers with 32, 64 and 128 neurons. In 2 hidden layers with 128 neurons achieves the peak accuracy of 0.888. In Fig.3.8, increasing hidden layers lowers performance as accuracy under 2 hidden layers out-weights 4 and 6 hidden. Besides these, larger neurons degrade accuracy especially in 4 and 6 layers. For instance, Fig.3.8c demonstrates 6 hidden layers without PCA accuracy which drops from 0.858 with

32 neurons to 0.786 with 128 neurons. It is worth to note that 0.786 is the minimal average accuracy in Fig. 3.8.

Fig.3.9 shows a DBN performance of 2, 4 and 6 hidden layers with different neurons numbers in each layer and accompanying the PCA method. The best average performance is achieved in 64-64 with 0.923. In Fig.3.9a, the values of average accuracy, with 0.918, 0.923 and 0.922 for 32, 64 and 128 neurons respectively are close to each other and outperforms the performance under 4 and 6 hidden layers. Fig3.9 demonstrates increasing the number of hidden layers decreases performance gradually. More specifically, with 128 neurons, the average accuracy degrades from 0.918 in 2 layers to 0.91 in 4 layers and finally 0.892 in 6 layers. Besides, smaller number of neurons perform better accuracy; especially in 4 and 6 hidden layers, 32 neurons outperform the detection performance with 64 and 128 neurons. The lowest value (0.892) is experienced in 6 layers with 128 neurons.

In order to illustrate different number of neurons impacting performance clearly, Fig. 3.8 and Fig. 3.9 apply different scales. Meanwhile, PCA impacts performance obviously so the same exact scale is not used in Fig. 3.8 and Fig. 3.9 to maintain readability. When Fig. 3.8 and Fig. 3.9 are compared, PCA appears to smooth fluctuations and improves performance. Fluctuation is expected and cannot be eliminated in a deep neural network. Fig. 3.8 and Fig.3.9 present 10 individual runs for detection. With PCA, total runs exhibit an accuracy of 0.94 to 0.85. As shown in the Fig. 3.8, PCA-preceded DBN does not only increase the average accuracy but also momentary accuracy. The average values which are obtained under 10 runs of DBN are summarized in Table 3.4 which indicates that DBN with PCA produces outperforms the architecture without PCA.

Result analysis

Numerical results show that, data scarcity and imbalance of legitimate and illegitimate classes can be overcome by Synthetic Minority Oversampling, and the training performance can be improved by introducing a PCA module prior to the DBN network. The results verify that PCA applied DBN structure having 2 hidden layers with 64 neurons provide the better accuracy than others. The proposed frameworks can achieve up to 92.3% accuracy, 94.2% precision, and 92.8% F1 score outperforming the DBN implementation without dimensionality consideration and even under the conditions of highly available sensing task submission data.

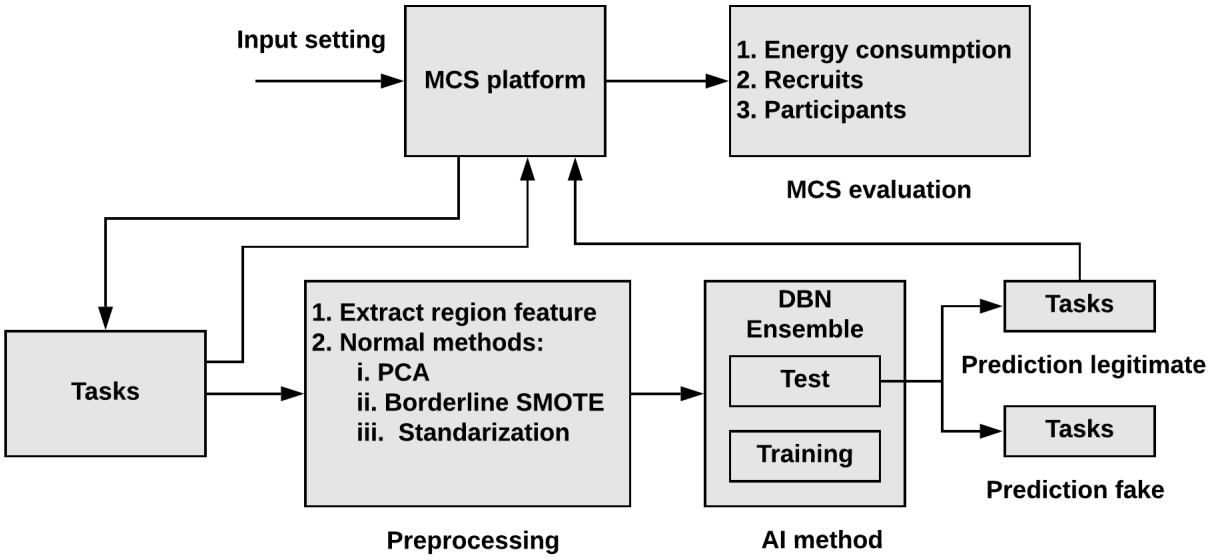


Figure 3.10: K-means-integrated system overview.

3.3 K-means-based Feature Extraction Integrating Bagging and Deep Learning-based Fake Task Detection in MCS Platforms

3.3.1 System Overview

AI-based fake task detection methods are proposed on an MCS server to mitigate fake tasks before assigning them to participants. MCS platform performance depends on AI-based detection method so energy consumption, recruits and participants are evaluated in the MCS system. Based on the analysis, the system is presented in Fig. 3.10 which demonstrates MCS platform generating tasks including fake tasks and legitimate tasks based on specific input parameters. The generated tasks make up the dataset to be used for training of the machine/deep learning models. Furthermore, preprocessing is applied since it is an essential step for AI algorithms to obtain the target performance. After dataset preprocessing, AI method trains itself using the training dataset and predicts legitimacy of each task in the test dataset. AI-driven techniques protect the MCS platform by predicting and excluding fake tasks. It means prediction of legitimate tasks are sent to the participants rather than all tasks as presented in the system overview in Fig. 3.10.

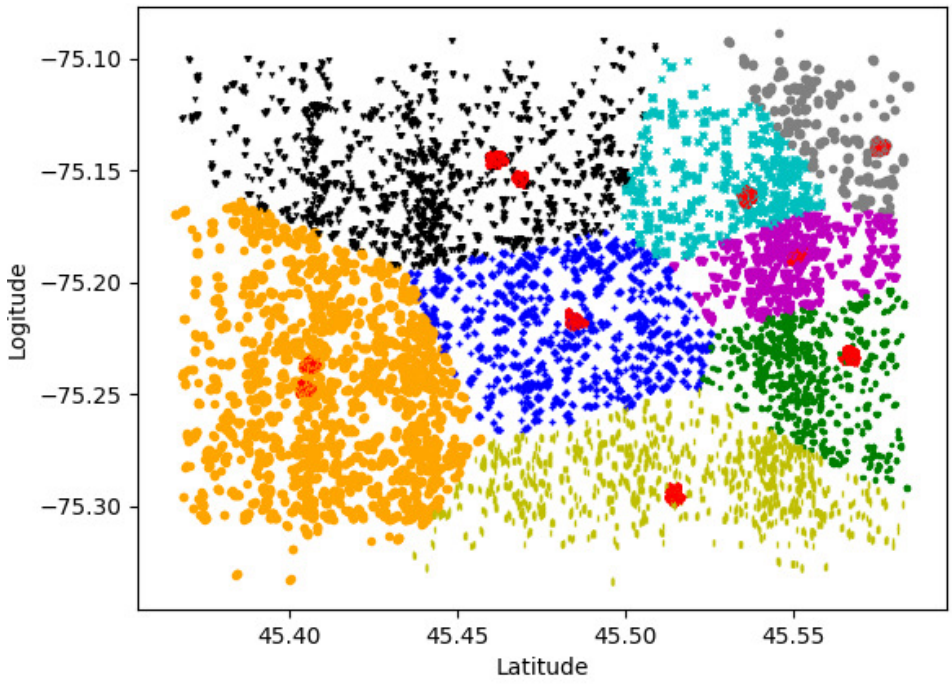


Figure 3.11: Dataset classified into eight areas by k-means. Red points denote fake tasks. All other colors indicate legitimate tasks that are split into eight areas.

3.3.2 Data Preprocessing and introducing region-awareness

Data preprocessing is to process features of data so that algorithms can explain it simply. As stated in [98], preprocessing data by extracting features from the original dataset boosts neural network-based classifier performance. An additional region feature is extracted upon an attack zone by k-means algorithm [118] and deploy prevalent and standard preprocessing approaches including Principal Component Analysis (PCA) [58] for dimensionality reduction, Borderline SMOTE over-sampling method [85] to tackle imbalanced dataset problem and standardization. These preprocesing methods are common and widely used.

Fake tasks concentrate on specific zones as mentioned before. Hence, one hypothesize is that the attack region is a significant feature. Here, k-means is applied to classify tasks based on 'latitude' and 'longitude' features as formally presented by Algorithm 1. K-means is easy to understand and implement. It's a straightforward and intuitive clustering algorithm. Meanwhile, k-means is computationally efficient [97]. Thus k-means algorithm is selected in this thesis. According to the algorithm, fake tasks are initially clustered by the k-means algorithm. Following upon that, centroids of fake tasks are considered as the initial centroids for the next step. This is the most important step to adapt k-means to our setting so to extract the attacked region information. Finally, the whole dataset is clustered by the k-means algorithm. Fig.3.11 demonstrates the classification result that shows each cluster contains only one attack zone. There are eight attack zones determined based on two task attributes: 'latitude' and 'longitude'. In the numerical evaluation, the number of clusters (k) in Algorithm 1 is set at eight, that is based on analyzing the original dataset characteristics. The original dataset consists of eight attack zones. Therefore, k is set as eight. Cluster results with specific numbers are marked as the extra features to be fed into the training models.

Algorithm 1: K-means-based task clustering regarding task attributes 'latitude' and 'longitude'.

Input:
k (the number of clusters);
D = DF + DL (D:set of total tasks, DF: set of fake tasks, DL: set of legitimate tasks);
Output: a set of k clusters

Cluster fake tasks by k-means:
arbitrarily choose k objects from DF as the initial centroids;
repeat
    (re)assign each object in DF to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster;
    update the cluster centroids;
until *convergence criteria is met or maximum number of iterations reached*;
Cluster all tasks by k-means:
get the centroids of fake tasks clusters as initial centroids;
repeat
    (re)assign each object in D to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster;
    update the cluster centroids
until *convergence criteria is met or maximum number of iterations reached*;

3.3.3 AI methods for fake task detection

In our previous work [51], DBN demonstrated encouraging performance in binary classification so it is worthwhile to continue investigating it. In addition, the study in [272] presents two ensemble learning-based approaches: random forest and gradient boosting to mitigate illegitimate tasks and obtain promising performance. Besides these, Bagging is an efficient machine learning method under the ensemble approaches [222] which often considers homogeneous weak learner, learns them independently from each other in parallel and combines them to determine the averaging process. However Bagging is rarely utilized to identity fake tasks in MCS systems. As an ensemble method, Bagging has potential to result in improved performance [272]. Therefore, in this thesis Bagging are deployed in an MCS server to detect malicious tasks in the system with motivation to boost detection performance. Meanwhile, DBN is built to compare with Ensemble learning performance as illustrated in Fig. 3.10. Furthermore, the features of the dataset impact DBN/Bagging training and test performance remarkably. Thus, DBN/Bagging-based task classification

performance depends on region feature to some extent. Therefore, experiments and numerical results demonstrate how region-awareness affects AI-based fake tasks mitigation in Section 3.3.4.

3.3.4 Performance Evaluation

In this evaluation section, DBN and Bagging algorithms are used to evaluate performances of mitigating illegitimate tasks. Hereafter, numerical results of MCS platform in terms of user recruitment, energy consumption and participants are presented. Moreover, it is impossible to reach 100% precision of the AI method while its side effect to the MCS platform is also present in the end.

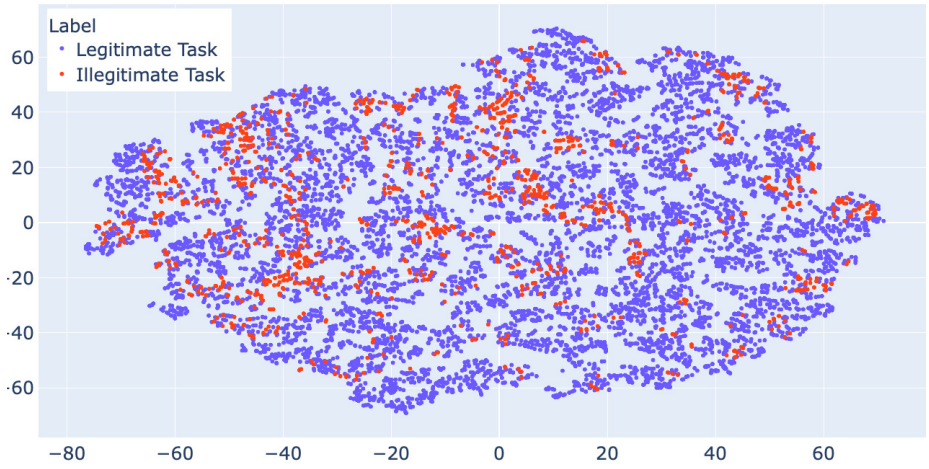
CR dataset is used which is generated by CrowdSenSim as introduced in 3.1.2. In the recruitment scenario, $10,000$ users with smart phones walk around the city Clarence-Rockland randomly. In addition, the probability of the battery level of a mobile device to be between 20% to 100% is 80% whereas with a probability of 20%, the initial battery level of a mobile phone is lower than 20%.

Table 3.5: DBN training accuracy comparison under different configuration. Proposed configuration with 32 neurons, 0.05 learning rate (LR), 1000 epoch. Acc:Accuracy. Conf:Configuration

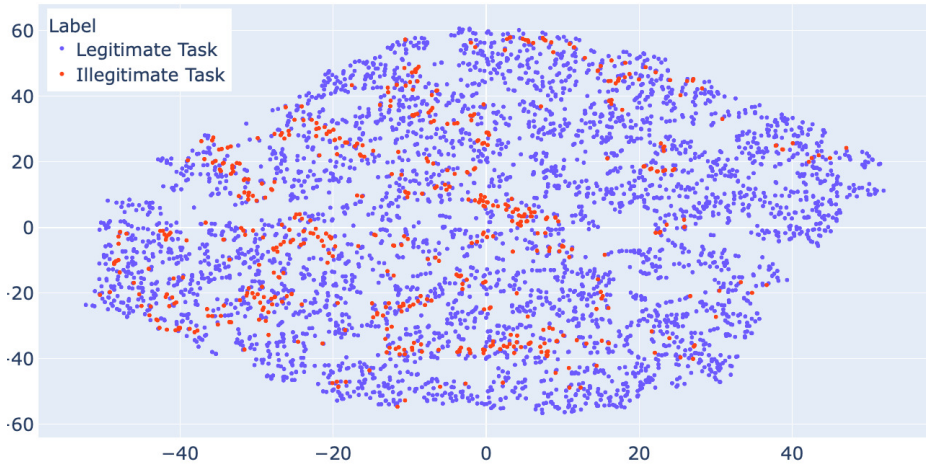
Conf	Proposed	Neurons 16	LR 0.02	LR 0.08	Epoch 800	Epoch 1200
Acc	0.985	0.954	0.956	0.975	0.967	0.984

Table 3.6: DBN and Bagging performance with/without region feature comparison.

	DBN	DBN & region	Bagging	Bagging & region
Precision	0.971	0.979	0.995	0.994
Recall	0.966	0.977	0.997	0.997
F1	0.966	0.977	0.996	0.996
Accuracy	0.966	0.977	0.993	0.993
G-mean	0.962	0.975	0.982	0.980



(a) Training dataset.



(b) Test dataset.

Figure 3.12: Dataset visualization using t-SNE. Red points represent illegitimate tasks and blue points for legitimate tasks.

Performance of DBN and Bagging Fake Tasks Detection

As presented earlier, k-means algorithm partitions the dataset into eight independent regions and each region is denoted by an integer, and provided as an additional feature of the dataset to the training model. It is worth to note that seven attributes are chosen in the tests so format of a tested task is as follows = {'latitude', 'longitude', 'hour', 'duration', 'resource', 'onpeakhour', 'gridnum'}. The selection is based on previous studies in

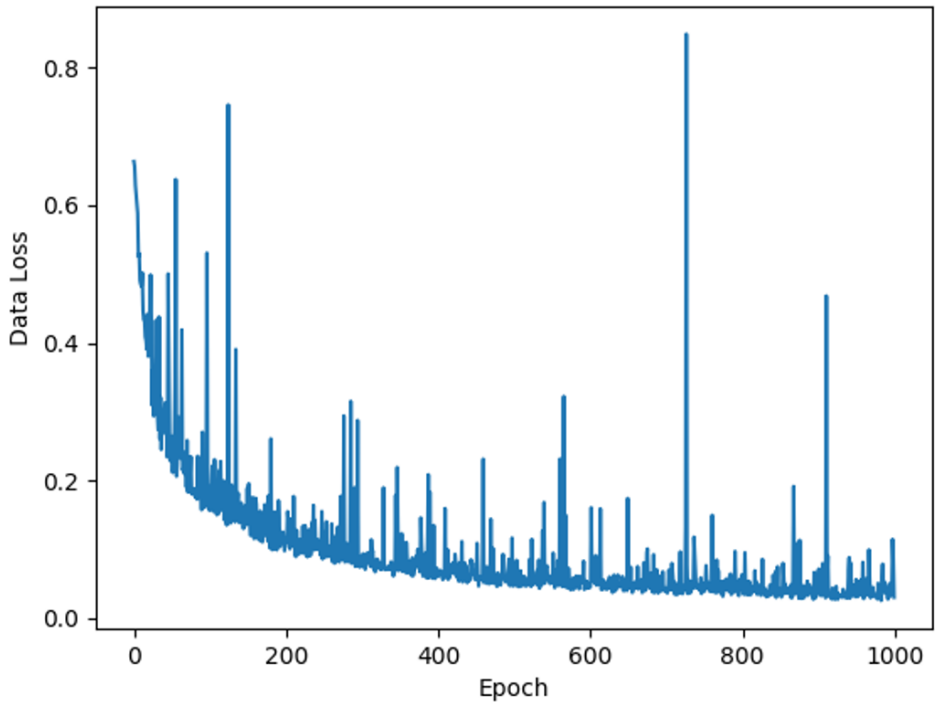


Figure 3.13: Epoch vs. data loss under DBN training process with configuration under 2 hidden layers, 32 neurons in each layer and 0.05 learning rate. Dataset without the region feature.

[218, 51] that AI-based method obtains promising prediction performance. Dataset is split into training (70%) and testing (30%) partitions. In order to visualize multi-dimensional datasets, t-distributed stochastic neighbor embedding (t-SNE) has become widespread in the field of machine learning [154, 192] . Briefly, t-SNE outputs two-dimensional "maps" from data with multiple dimensions that can be even at the order of hundreds. Fig. 3.12 uses t-SNE to visualize the training and test datasets that consist of seven features. The testing dataset consists of a total 4,346 tasks: 3,780 legitimate tasks and 566 fake tasks. Parameters of the DBN model are as follows: (1) 2 hidden layers, (2) 32 neurons in each layer, (3) learning rate: 0.05 and (4) epoch: 1000. In addition, DBN training accuracy is compared under proposed configuration and others in Table.3.5 which shows that the DBN training accuracy performs best under our selected parameters. Furthermore, Fig. 3.13 demonstrates DBN data loss reduce gradually and reach the minimum possible value in the training process under this configuration. It means DBN converges gradually and results in 0.985 training accuracy. Regarding the Bagging learning method, a Decision

Tree is selected as the base estimator and 10 estimators in it. The average values which are obtained under 10 runs of DBN and Bagging in terms of precision, recall, F1-score and G-mean with and without the region feature are summarized in Table 3.6. As mentioned in Sec 4.2.2, the dataset used is imbalanced so geometric mean (G-mean) [114] together with other four measurements(precision, recall, F1 and accuracy) are evaluated. G-mean score aims to maximize the accuracy of each of the classes while keeping these accuracy levels balanced. As a result, it is usually utilized to evaluate machine learning algorithm performance under imbalanced datasets. For binary classification, G-mean is formulated as the square root of the product of the sensitivity and specificity as shown in Eq. (3.1).

$$G - mean = \sqrt{sensitivity * specificity} \tag{3.1}$$

,where

$$sensitivity = \frac{TP}{TP + FN}$$

,

$$specificity = \frac{TN}{FP + TN}$$

TP, TN, FP and FN stand for true positive, true negative, false positive and false negative.

Table 3.6 demonstrates that Bagging produces improved prediction results when compared to DBN in both with and without region feature. However, region feature reduces Bagging performance marginally, in terms of G-mean from 0.982 to 0.980 and in terms of precision from 0.995 to 0.994. On the other hand, area feature improves the DBN performance slightly in terms of precision from 0.971 to 0.979, in terms of recall/F1/accuracy from 0.966 to 0.977 and in terms of G-mean from 0.962 to 0.975. Although the region feature causes a very slight reduction on Bagging-based classification performance, it proves to be an effective feature by boosting DBN performance.

Impact of fake task mitigation

Performance of DBN and Bagging varies under region-aware and region-unaware settings. Here, the region-aware bagging and DBN approach is used to evaluate the impact of fake task mitigation on the MCS platform. The simulation results illustrate energy consumption under the region-aware DBN and region-aware Bagging in comparison to a baseline scheme where no mitigation strategy is introduced (see Fig. 3.14). Energy consumption indicates the sum of battery unit percentage of all recruits drained for the total recruitment. For example, if a fake task needs 7% battery level and another fake one requires 8%, then the

AI-based Prevention of Battery-oriented Illegitimate Task Injection in MCS consumption of these two tasks will save 15% battery energy. Meanwhile, tasks without participants are excluded from the energy statistics calculation since they do not drain resources for the MCS campaign. Fig.3.14 illustrates fake tasks' energy consumption and cut slices demonstrate energy (i.e. battery lifetime) saving when Bagging/DBN is applied. Thus, Bagging saves 12.18% of the battery energy, which is slightly higher than DBN with a saving of 11.81%.

Besides, AI-based mitigation of fake tasks protects the majority of MCS players from the battery-oriented illegitimate task submissions. The impacted recruits and participants are formulated in Eq. (3.2) and Eq. (3.3), respectively.

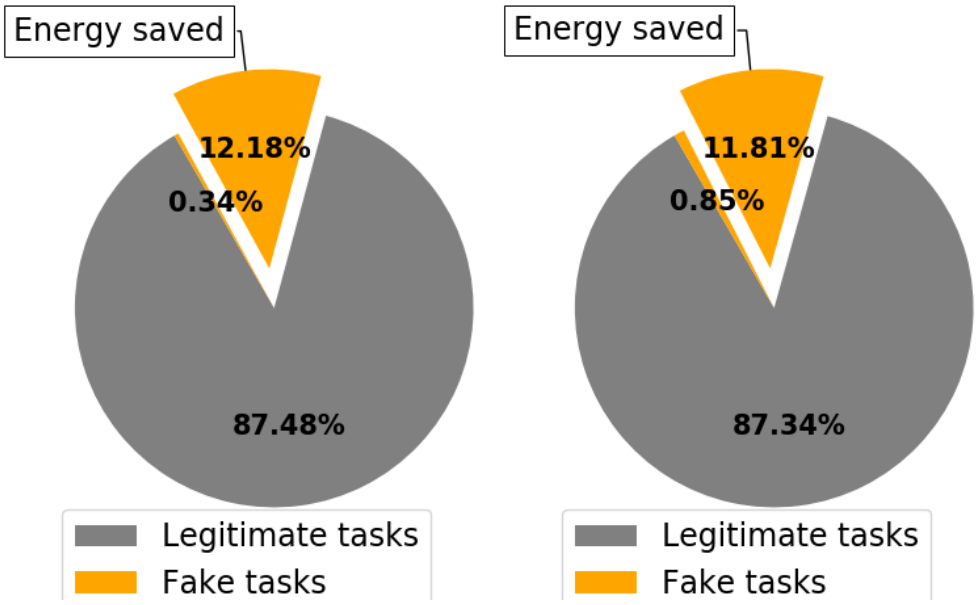
$$impacted_{recruits} = \frac{FT\ recruits}{Total\ recruits} \tag{3.2}$$

$$impacted_{participants} = \frac{Participants\ in\ FT}{Total\ participants} \tag{3.3}$$

FT stands for the set of fake tasks in Eq. (3.2) and Eq. (3.3). It is possible to have FP prediction results which means illegitimate tasks are predicted as legitimate tasks, and thus, those illegitimate tasks are allowed to recruit participants in MCS activities. Table 3.7 and Table 3.8 reports the average of ten run test results by comparing the DBN and Bagging-based results to a task assignment scenario where there exists no fake task detection mechanism (i.e., denoted W/N AI in the tables). Table 3.7 reports that the number of fake task recruits reduces sharply to 1 and 2.8 when Bagging and DBN applying respectively. In addition, it demonstrates the impacted recruits drop sharply, from 10.56% to 0.25% when Bagging is applied. While DBN remedies fake tasks recruits from 10.56% to 0.71%. Moreover, the proposed methods results in participants in fake tasks declining dramatically from 342.7 to 9.9 when Bagging utilised and to 25.6 if DBN applied in Table 3.8. Moreover, the numerical results report that Bagging and DBN help to decrease the impacted participants dramatically to 0.35% and 0.91% respectively, in comparison to 10.94% under the 'W/N AI' task assignment as presented in Table 3.8. Thus, Bagging-based methodology secures 10.59% of the participant population while DBN secures 10.03% of the participant population from malicious sensing tasks.

Legitimate task completion rate

False negatives are inevitable. Therefore, AI methods' prediction accuracy can not be 100% regularly. It means Bagging and DBN methods may misclassify some legitimate tasks into



(a) Bagging (b) DBN

Figure 3.14: Energy savings under Bagging and DBN.

Table 3.7: Recruits for fake tasks and total of fake task versus legitimate tasks comparison (FT = fake task).

	FTs recruits	Total recruits	Impacted recruits
W/N AI	46.5	440.4	10.56%
Bagging	1	393.1	0.25%
DBN	2.8	392.9	0.71%

Table 3.8: Participants in fake tasks and in total of fake tasks and legitimate tasks comparison (FT = fake task).

	Participants in FTs	Total participants	Impacted participants
W/N AI	342.7	3133.3	10.94%
Bagging	9.9	2825.8	0.35%
DBN	25.6	2819.6	0.91%

Table 3.9: Task completion rate (TCR).

W/N AI	Bagging		DBN	
TCR	TCR	Influence rate	TCR	Influence rate
10.43%	10.37%	0.57%	10.32%	1.05%

the fake task cluster that are discarded by our proposed system in Fig. 3.10. Therefore, DBN and Bagging may result in some legitimate tasks unaccomplished. The legitimate tasks completion rate (TCR) is investigated, which is formulated in Eq.(3.4).

$$TCR = \frac{Totalrecruits - FTsrecruits}{Numoflegtasks} \tag{3.4}$$

, where $Totalrecruits$ denotes number of recruits, and $FTsrecruits$ stands for number of fake task recruits. Both of these appear in Table 3.7, and Numoflegtasks denotes number of legitimate tasks, which is 3,780 as mentioned earlier. Table 3.9 compares the legitimate task completion without AI (i.e. baseline), to Bagging and DBN. The results show that Bagging and DBN reduces TCR marginally, from 10.43% to 10.37% and 10.32% respectively. The

paper [274] shows the influence rate calculation formulation in Eq. (3.5)

$$Influencerate = \frac{TCRw/oAI - TCRwithAI}{TCRw/oAI} \tag{3.5}$$

Influence rate for Bagging and DBN is calculated as 0.51% and 1.05%, respectively by using Eq. 3.5 based on the obtained numerical TCR results. Thus, Bagging and DBN result in 0.51% and 1.05% legitimate tasks unfinished, which is a significantly low side effect in MCS campaigns.

3.3.5 Result analysis

Incorporation of region-awareness into DBN has improved the overall accuracy from 0.966 to 0.977 and G-mean from 0.962 to 0.975. Both DBN and Bagging demonstrates competitive performance on binary identification of legitimate and illegitimate tasks. More specifically, Bagging performs up to 0.995 precision, 0.997 recall, 0.996 F1, 0.993 accuracy and 0.982 G-mean. Thus, majority of the fake tasks have been detected and eliminated before being assigned to MCS participants. As a result, Bagging leads to 12.19% energy saving for the participants and reduces impacted recruits dramatically to 0.25%, when compared to 10.56% under a baseline approach with no intelligent strategy for fake task mitigation. Furthermore, Bagging decreases the ratio of participants impacted by fake tasks from 10.94% to 0.35%.

3.4 Federated Learning-Based Decentralized System to Mitigate Fake Task Impacts on Crowdsensing Platforms

3.4.1 System Overview

An alternative to a centralized approach to detect the fake submissions is to decentralize the decision and offload detection to the participating devices, which yields FL. This approach also mitigates the concerns about privacy in data provisioning to the centralized servers' ML models. FL is a methodology integrating a distributed ML paradigm that trains a shared and dispensed artificial intelligence (AI) model using local training datasets on distributed devices [255]. Research on the application of FL to fake task detection is

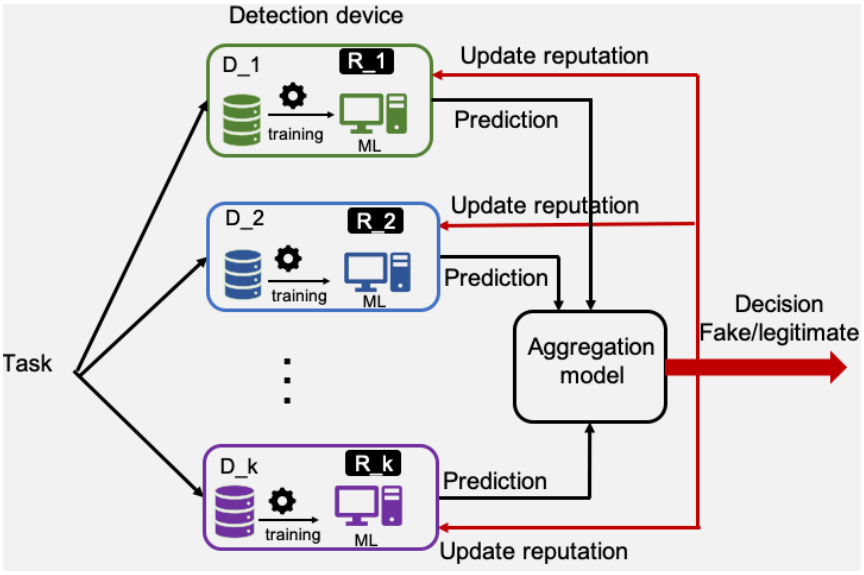


Figure 3.15: Federated learning based fake task detection system

limited. Motivated by the state of the art in fake task detection and the merit of FL techniques, to the best of our knowledge, for the first time, a FL-based framework is proposed to identify and mitigate illegitimate tasks in MCS platforms. A novel horizontal FL-backed framework is introduced to identify malicious tasks in MCS systems. The proposed system is comprised of various detection devices and a risk/loss-aware[1] aggregation module. The aggregation module can be deployed in an MCS platform in the cloud. A detection device runs a local ML model that may vary from one device to another (e.g., Bagging, Adaboost and Extreme Gradient Boosting (XGBoost)) and a local dataset. The ML model is trained by the local dataset in the device and delivers prediction results independently. The aggregation module is responsible for gathering decisions from devices and making a concluding decision. Bayesian decision theory suggests making statistic-driven decisions with the objective of maximizing the expected utility over a model posterior [122]. With this in mind, an aggregation module is designed that makes decisions subject to the minimum utility loss over the actions regarding the classes of incoming MCS tasks. The FL-based framework demonstrates not only benefits of illegitimate tasks detection, but also promising detection performance.

[1]We use risk-aware and risk/loss-aware interchangeably as risk is the cost of utility losses due to misclassifying a task under the wrong legitimacy class and losing a legitimate task value or incurring the cost of an illegitimate task.

Rather than current features for a task as introduced in Section 3.1.1, a new feature is added, namely 'TaskValue', for tasks assigned an integer value uniformly from the range 1 to 10. 'TaskValue' is used for calculating loss rather than training ML models. As 'TaskValue' is a feature of a sample point, it can be obtained by task 'ID'.

Reputation-aware Federated Learning Model

The proposed FL-based detection system deploys several distributed devices with a local dataset and a ML model, as illustrated in Fig. 3.15. Each device aggregates sensing tasks upfront and stores the dataset locally. The ML model in each device is trained using a local dataset before receiving a new sensing task. The MCS platform then distributes tasks submitted from end-users to all detection devices. The trained ML models in devices estimate a task label. In an aggregation module, a risk-aware method is utilized to quantify the loss of all possible actions in the decision-making stage. Before introducing risk-awareness, the initial aggregated legitimacy decision for $task_i$ (S_i) is calculated as $S_i = \sum_{j=1}^{k} DS_i^j$ where the decision of $device_j$ for $task_i$ (DS_i^j) is weighted by the reputation of the device at the end of the previous aggregated decision for $task_{i-1}$ as in (3.6) where ES_i^j stands for the local ML model-based prediction value of $device_j$, and R_{i-1}^j is $device_j$ reputation after $task_{i-1}$. Fake tasks and legitimate tasks are labeled with the 0 and 1, respectively. Based on (3.6), the initial aggregated legitimacy can be re-formulated as in (3.7).

$$DS_i^j = ES_i^j R_{i-1}^j \tag{3.6}$$

$$S_i = \sum_{j=1}^{k} ES_i^j R_{i-1}^j \tag{3.7}$$

Device reputation is updated upon every aggregated decision on task legitimacy. Reputation of $device_j$ after $task_i$ is calculated as in (3.8) where CP_i^j and CD_i^j are used to adjust the reputation of $device_j$. The two parameters start with 0 so devices are assigned an initial reputation of 0.5. If the local ML model of $device_j$ outputs the same result as the aggregation module, both CP_i^j and CD_i^j are incremented by one. Otherwise, the value of CP_i^j remains while CD_i^j is incremented. Increasing CP_i^j will favor the reputation of $device_j$, whereas an increase in CD_i^j could drag reputation down. In an ideal situation, the decision of $device_j$ always matches the aggregated decision, and after N task decisions, the values of CP_N^j and CP_N^j reach $N-1$.

$$R_i^j = \begin{cases} \frac{\epsilon + CP_i^j}{2 \times \epsilon + CD_i^j} & i \geq 2 \\ 0.5 & i = 1, CP_i^j = CD_i^j = 0 \end{cases} \tag{3.8}$$

When R_i^j and ϵ denote the reputation of $device_j$ after predicting $task_i$ legitimacy and a small value for device reputation calculation, respectively, according to (3.8), reputation is calculated as follows for an always correct device:

$$R_N^j = \frac{\epsilon + (N-1)}{2 \times \epsilon + (N-1)} \tag{3.9}$$

where N is total number of sensing tasks. Since ϵ is a negligibly small value, the limit of (3.9) approaches 1 as expressed in (3.10).

$$\lim_{N \to \infty} R_N^j = \lim_{N \to \infty} \frac{\epsilon + (N-1)}{2 \times \epsilon + (N-1)} = 1 \tag{3.10}$$

Thus, reputation values for all devices should be less than 1 due to the limitation of total task number N and almost impossible 100% matching with aggregation module. According to (3.7) and (3.8), the reputation-aware aggregation values can be formulated by (3.11) where k denotes total number of detection devices.

$$S_i = \begin{cases} \sum_{j=1}^{k} ES_i^j \frac{\epsilon + CP_i^j}{2 \times \epsilon + CD_i^j} & i \geq 2 \\ \frac{1}{2} \sum_{j=1}^{k} ES_i^j & i = 1 \end{cases} \tag{3.11}$$

Equation (3.11) defines the aggregation results of $task_i$ that relies on the ML-based estimation value ES_i^j and the device reputation. As introduced before, ES_i^j is either 0 or 1 representing malicious tasks and legitimate tasks, respectively. If $device_j$ estimates $task_i$ as fake, ES_i^j is set to 0. In this case, devices predicting $task_i$ as illegitimate do not contribute in updating S_i. Accordingly, (3.11) formulates a summary of legitimate prediction for $task_i$. Based on these analyses, the overall probability of task prediction to be legitimate is formulated by (3.12).

$$P(l|T_i) = \frac{S_i}{k} \tag{3.12}$$

$P(l|T_i)$ represents probability of prediction $task_i$ as legitimate, and $P(f|T_i)$ denotes the probability of predicting $task_i$ as fake. Meanwhile, $P(l|T_i)$ and $P(f|T_i)$ should satisfy the condition: $P(l|T_i) + P(f|T_i) = 1$. According to this condition and (3.12), probability of predicting $task_i$ as fake is obtained in (3.13).

$$P(f|T_i) = 1 - \frac{S_i}{k} \tag{3.13}$$

Table 3.10: Utility loss description

	True Legitimate	True Fake
Predict Legitimate	0	λ_1
Predict Fake	λ_2	0

Table 3.10 presents the utility loss due to classification decisions. Specifically, the ML model predicting a fake task as fake or predicting a legitimate task as legitimate impacts the MCS platform's utility. On the other hand, incorrect estimation results in a loss. Thus, λ_1 is the loss due to classifying a fake task as legitimate whereas λ_2 is the loss incurred due to classifying a legitimate task as fake. Therefore, each decision action entails a certain risk, and the risks of legitimate and fake predictions are formulated respectively as follows:

$$RT_i = P(f|T_i) \times \lambda_1 \times TV_i \tag{3.14}$$

$$RF_i = P(l|T_i) \times \lambda_2 \times TV_i \tag{3.15}$$

Given that RT_i and RF_i stand for the risk of predicting $task_i$ as legitimate and fake, respectively whereas TV_i denotes value of $task_i$, (3.14) defines the risk of predicting $task_i$ as legitimate and Equation (3.15) formulates the risk of predicting $task_i$ as fake. Aggregation module determines the legitimacy of a task according to a risk value. Specifically, risk is a function of utility loss due to a taken action as formulated in (3.16) where $FD(T_i)$ denotes the final aggregated decision for $task_i$. Thus, the final decision chooses the least risky action.

$$FD(T_i) = \begin{cases} 1 & RT_i \leq RF_i \\ 0 & RT_i > RF_i \end{cases} \tag{3.16}$$

Vote based federated learning system

With the concept of static reputation value for devices, a vote-based aggregation approach is designed in the FL platform as an alternative to the reputation-based approach. Thus, R_i^j is set to a constant value of α that remains the same for all tasks. Then aggregation value in (3.7) can be re-formulated as follows:

$$S_i = \sum_{j=1}^{k} ES_i^j \times \alpha \tag{3.17}$$

As introduced in Section 3.4.1, a device's reputation should not exceed 1. As a result, α should be in the range of 0 to 1. The device's decision is eliminated by the aggregation module setting $\alpha = 0$, whereas $\alpha = 1$ stands for completely accepting the device's decision based upon (3.17). Meanwhile, devices are partially trusted when α is less than 1. To reduce the computational complexity, α is chosen as 1. Probability of predicting $task_i$ as legitimate is formulated as shown in (3.18):

$$P(l|T_i) = \frac{\sum_{j=1}^{k} ES_i^j}{k} \tag{3.18}$$

According to (3.13) and (3.18), $P(f|T_i)$ is deduced. Loss calculation and aggregation decision rules are the same as reputation-aware FL model showing (3.14), (3.15) and (3.16), respectively.

3.4.2 Performance Evaluation

Based on the design of tasks, the CrowdSenSim tool generates the dataset using real physical features (e.g., using Timmins, a small city in Canada as the simulated setting [72]. It is an imbalanced dataset with 89% legitimate tasks and 11% fake tasks. The dataset is composed of $1,000$ MCS tasks in total, as noted in our prior work [50]. In [50], 800 tasks are used for training a DBN model and 200 tasks are used for testing. DT_0 is a training dataset the same as in [50]. DT_0 is uniformly split into two small datasets DT_1 and DT_2 under constraints: 1) size of DT_1 and DT_2 are both half the size of DT_0; 2) $DT\_0 = DT\_1 \cup DT\_2$; 2) $DT\_1 \cap DT\_2 = \emptyset$. Here, the same test dataset is applied to make fair comparisons to the previous results.

Comprehensive experiments are executed to verify performance of mitigating illegitimate tasks based on different configurations including 1) reputation (e.g., dynamic and static), 2) loss metric (e.g., λ_1 and λ_2), and 3) the size of the local dataset saved in devices (e.g., 800 tasks and 400 tasks). Meanwhile, the proposed FL outperforms the centralized system detection results performed by one ML model. Considering system complexity, the number of detection devices k should not be a large value. k is set up as 5. Meanwhile, ϵ is set as $(1.0e-5)$.

Centralized test platform performance

The study in [50] demonstrated a Deep Belief Network (DBN)-based fake task detection system in MCS. Several ensemble algorithms such as XGBoost [45], AdaBoost[207] and

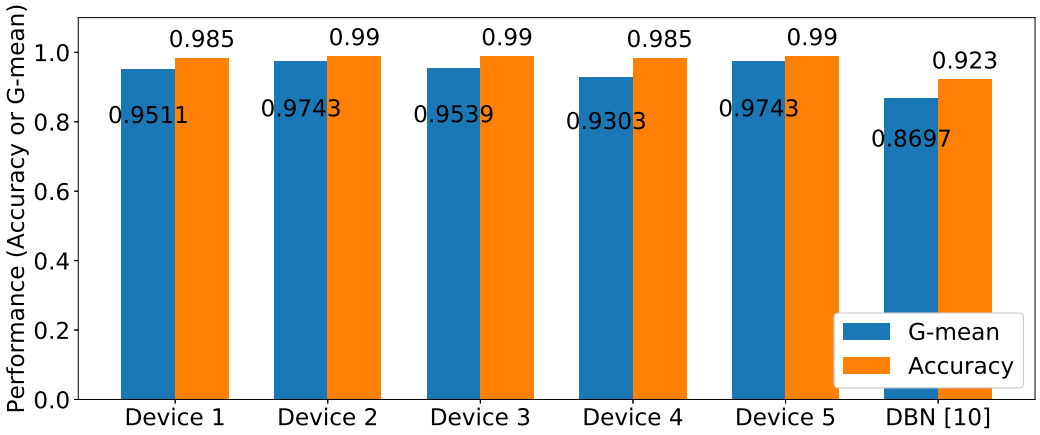


Figure 3.16: Centralized performance

Table 3.11: Performance by centralized system

Device	Precision	Recall	F1	G-mean	Accuracy
1	0.9848	0.9850	0.9848	0.9511	0.9850
2	0.9900	0.9900	0.9900	0.9743	0.9900
3	0.9901	0.9901	0.9898	0.9539	0.9900
4	0.9852	0.9850	0.9845	0.9303	0.9850
5	0.9900	0.9900	0.9900	0.9743	0.9900
DBN [50]	0.9420	0.9230	0.9280	0.8697	0.9230

Bagging [34] are used to mitigate malicious tasks. In centralized case, the aggregation module is disabled in the proposed system Fig. 3.15. The centralized test model configuration is shown in Table 3.12. The training dataset and test dataset are the same as in [50]. Table 3.11 presents prediction results. DBN results is referred in [50]. According to the results, all ensemble models boost detection performance dramatically than DBN [50]. Fig.3.16 illustrates G-mean and accuracy comparison results. It shows XGBoost in device 5 and Adaboost in device 2 demonstrate the highest G-mean achieving over 12% improvement over DBN.

Federated learning-based system

Through deploying different machine learning models and different local datasets in devices, two FL-based models are designed. FL MODEL_1 configuration is the same as centralized

system in Table 3.12. In this model, 800 samples are saved in all devices in advance. The FL MODEL_2 configuration shows in Table 3.13. In this model, only 400 data points are distributed to devices. Loss parameters λ_1 and λ_2 in Table3.10 are configured three different groups: (1)λ_1=1, λ_2=2; (2)λ_1=1, λ_2=1, and (3)λ_1=1, λ_2=0.5. Furthermore, both reputation aware and vote base approach with static reputation are considered. Table 3.14 presents MODEL_1 performance and average loss value. Table 3.14 shows the best detection performance in group (3) (λ_1=1, λ_2=0.5) for both dynamic and static reputation cases with up to 0.99 accuracy. From increasing the value of λ_2 boosts platform performance as well as marginally decreases average loss for both dynamic reputation and vote-based cases. Specifically, λ_2 increasing from 0.5 to 2 in dynamic reputation case, gives G-mean value improvement from 0.9303 to 0.9743 and loss per device decreases from 0.59 to 0.56. In the vote based case, the trend of loss and performance is the same as the dynamic reputation case. Raising λ_2 boosts detection accuracy and lowers loss. Fig.3.18 illustrates accuracy and G-mean comparison with different λ_2 values and reputation determination rules (i.e., vote-based or dynamic assessment) when 800 samples are fed in detection devices.

Table 3.12: Centralized model and FL-based model (MODEL_1) configuration. Num of BES denotes the number of base estimators.

Device	ML	Base estimator	Num of BES	Dataset
1	Bagging	Xgboost	100	DT_0
2	Adaboost	DecisionTreeRegressor	50	DT_0
3	Bagging	DecisionTree	100	DT_0
4	Bagging	RandomForest	100	DT_0
5	XGBoost	N/A	50	DT_0

Results in Table 3.14 describe average loss in static is critically lower than dynamic. The root cause is that the reputation is always less than '1' based on (3.10) if N is set to 200. It results in aggregation value S_i for $task_i$ in (3.11) should be less than 5 even if all devices predict this task as legitimate (5 devices model designed here). Therefore, probability of prediction task as legitimate in (3.12) should be always less than 1 and $P(f|T_i)$ always a positive value according to (3.13). It means prediction risk RT_i and RF_i are always a positive value. On the other hand, vote based system with static reputation 1 avoids loss when five devices estimate $task_i$ as legitimate. In this case, aggregation S_i is 5 according to (3.17). $P(l|T_i)$ is 1 in (3.18) and $P(f|T_i)$ is 0 in (3.13). With $P(f|T_i)$ 0, RT_i in (3.14) is 0. Fig.3.17 demonstrates loss of 200 test tasks with dynamic and static reputation respectively based on different loss metric parameters. From Fig. 3.17, most tasks loss in vote based MODEL_2 is 0 that contributes a smaller average loss than reputation aware

Table 3.13: Federated learning MODEL_2

Device	ML	Base estimator	Num of BES	Dataset
1	Adaboost	DecisionTreeRegressor	100	DT_1
2	Bagging	DecisionTree	50	DT_1
3	Bagging	DecisionTree	100	DT_2
4	XGBoost	N/A	100	DT_1
5	XGBoost	N/A	50	DT_2

MODEL_1.

Table 3.15 describes MODEL_2 performance and prediction loss. It shows the same trend as in MODEL_1 in terms of suppressing rate of λ_1 over λ_2 resulting in a reduction of loss and improvement of detection performance. In the case $(\lambda_1, \lambda_2) = (1, 2)$, the results are obtained inlcuind the least loss and 100% performance for both dynamic and static reputation considered cases. Two factors contribute to 100% accuracy and precision. Firstly, individual detection device performs encouraging detection accuracy showing in centralized detection model in Table 3.11. Secondly, the FL-based method relies on horizontal device decisions and gives different decisions via configuring loss metric parameters. Estimation DS_i^j of deployed five devices is 0, 0, 1, 1, 0 respectively. Aggregation result S_i is obtained as 2. According to (3.18), probability of legitimate is calculated as 0.4. After that, loss of predicting task as legitimate and loss is measured according to (3.14) and (3.15) using different loss parameters λ_1 and λ_2. When λ_2=2 λ_1, the aggregation module determines $task_i$ legitimate that is incorrect decision comparing with task label. If λ_1/ λ_2 is either to 1 or 2, FL-based model decides $task_i$ fake that is correct estimation. Therefore, adjusting λ_1/ λ_2, aggregation makes different decision of a task.

Comparing results in MODEL_1 (Table 3.14) and MODEL_2 (Table 3.15), MODEL_2 shows marginal improvement than MODEL_1 when λ_1/λ_2 is chosen as 1 and 2. More specifically, G-mean shows over 2.6% improvement from 0.9743 in MODEL_1 to 1 in MODEL_2 selecting λ_1/λ_2 as 2.

3.4.3 Result analysis

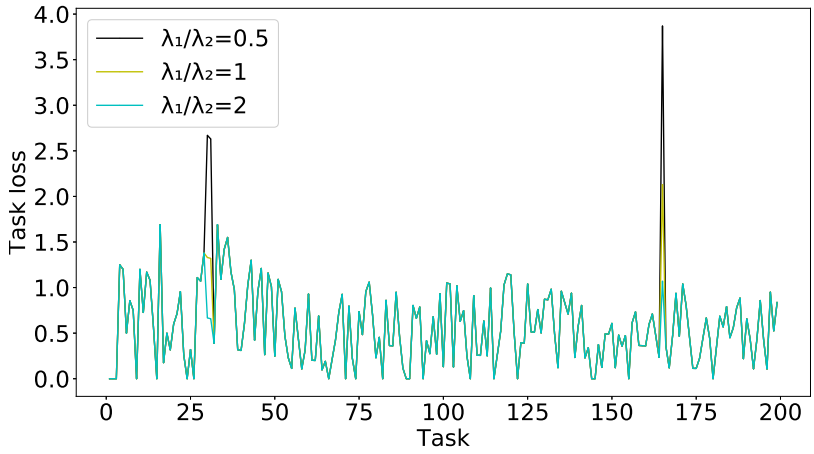
In the centralized system, experimental results show Adaboost and XGBoost perform the highest G-mean 0.9743 and accuracy 0.9900. In the FL-based model with a large dataset, the best performance is the same as a centralized system with G-mean 0.9743 and accuracy 0.9900 in dynamic and vote based cases. Under smaller datasets, the federated

Table 3.14: Performance in FL system MODEL_1

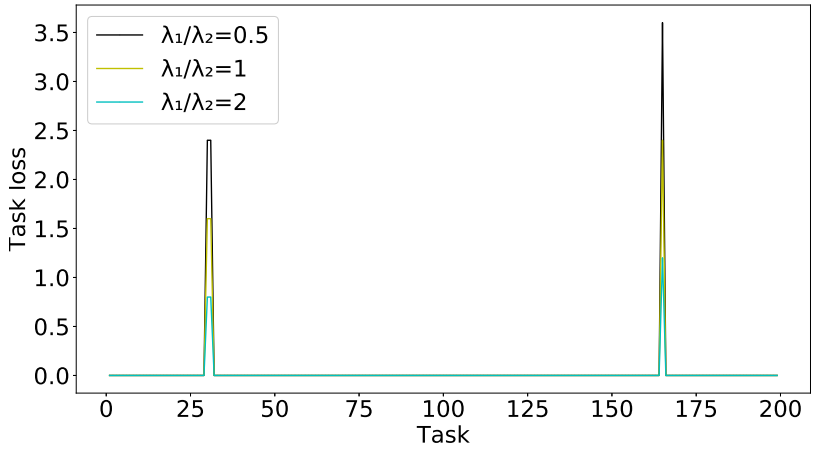
Config	Prec.	Recall	F1	G-mean	Acc.	loss
λ_1/λ_2=0.5 Dynamic	0.9852	0.9850	0.9845	0.9303	0.9850	0.59
λ_1/λ_2=1 Dynamic	0.9852	0.9850	0.9845	0.9303	0.9850	0.57
λ_1/λ_2=2 Dynamic	0.9900	0.9900	0.9900	0.9743	0.9900	0.56
λ_1/λ_2=0.5 Vote	0.9852	0.985	0.9845	0.9303	0.985	0.04
λ_1/λ_2=1 Vote	0.9900	0.9900	0.9900	0.9743	0.9900	0.03
λ_1/λ_2=2 Vote	0.9900	0.9900	0.9900	0.9743	0.9900	0.01

Table 3.15: Performance in FL system MODEL_2

Config	Prec.	Recall	F1	G-mean	Acc	loss
λ_1/λ_2=0.5 Dynamic	0.9757	0.9750	0.9736	0.8819	0.9750	0.63
λ_1/λ_2=1 Dynamic	0.9950	0.9950	0.9949	0.9771	0.9950	0.59
λ_1/λ_2=2 Dynamic	1.0000	1.0000	1.0000	1.0000	1.0000	0.56
λ_1/λ_2=0.5 Vote	0.9757	0.9750	0.9736	0.8819	0.9750	0.12
λ_1/λ_2=1 Vote	0.9950	0.9950	0.9949	0.9771	0.9950	0.08
λ_1/λ_2=2 Vote	1.0000	1.0000	1.0000	1.0000	1.0000	0.05



(a) Task loss with dynamic reputation



(b) Task loss with static reputation

Figure 3.17: Task value loss under dynamic reputation and static reputation in MODEL_1

model ensures up to 100% overall accuracy in both dynamic and static reputation cases with the loss incurred due to false prediction of legitimate and illegitimate tasks. Higher detection performance also leads to lower average loss in terms of task values for dynamic reputation and vote based federated models. The test results show that the key benefit of the proposed horizontal FL architecture is to boost fake task detection performance and reduce average loss.

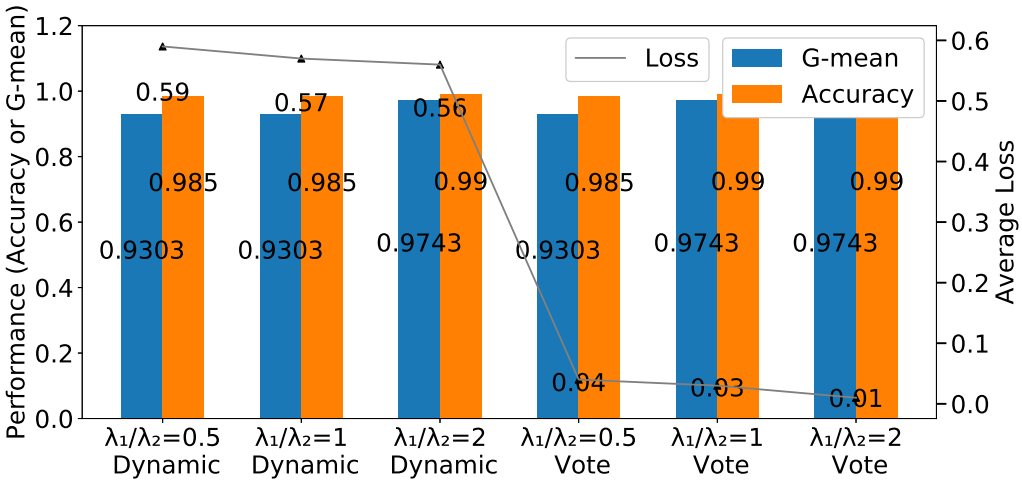


Figure 3.18: FL-based performance comparison in MODEL_1

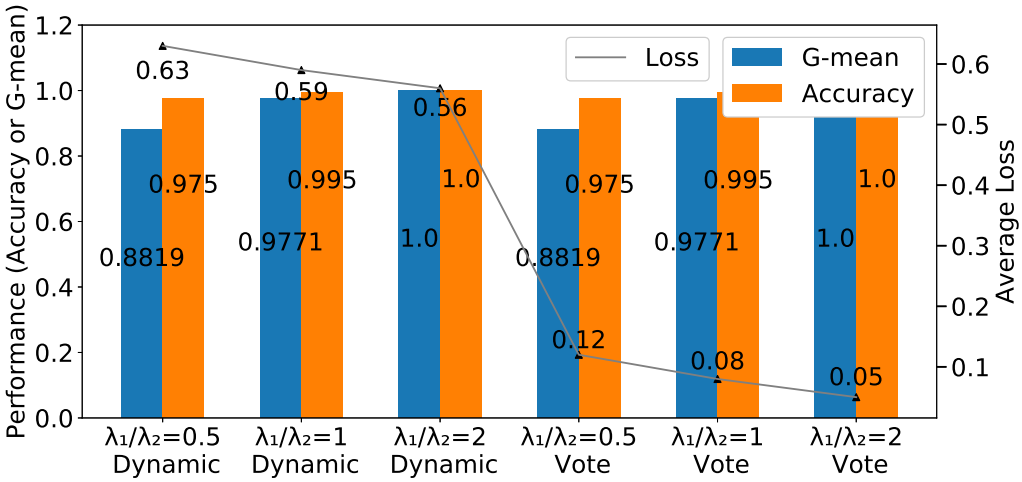


Figure 3.19: FL-based performance comparison in MODEL_2

3.5 Conclusion

MCS platforms are vulnerable to adversarial behavior such as data poisoning and illegitimate task injections. The former have been widely investigated whereas the latter still calls for effective solutions. Illegitimate task attack is one of the most essential threats that affects the performance of MCS campaigns due to draining the battery, sensing, memory and processing resources of participating devices and computing, communication and storage

resources at the sensing servers in the MCS platforms. As discussed before, this section introduces tasks characteristics (e.g., legitimate tasks and fake tasks) and tasks generation procedure firstly. After that, three methods for fake tasks detection are introduced, addressing particular issues in deploying different kinds of MLs in MCS platforms.

Firstly, DBN-based approach is proposed which aims for prevention of Artificial Intelligence driven fake (illegitimate) task submissions in MCS systems under the condition of scarce sensing task submission data. Through realistic MCS data generated by Crowdsensing on a real city map, the analysis is that the performance of a DBN preceded by a PCA module with various architectures. From numerical results, we can see oversampling technique overcomes data scarcity and imbalance between legitimate and illegitimate classes. The training performance can be enhanced by incorporating a PCA module prior to the DBN network. The results demonstrate that PCA-applied DBN structures with two hidden layers and 64 neurons give more precision than other structures. The suggested frameworks may achieve up to 0.923 accuracy, 0.942 percent precision, and 0.928 percent F1 score, exceeding the DBN implementation without dimensionality consideration and even with highly accessible sensing task submission data.

Secondly, the AI-based and region-aware strategy is proposed that builds on Deep Belief Network (DBN) and an ensemble machine learning algorithm (i.e., Bagging) against fake task injections to MCS platforms. To this end, k-means algorithm is integrated to cluster the task data so to extract region feature. The incorporation of region-awareness into DBN has increased the overall precision from 0.966 to 0.977 and the G-mean from 0.962 to 0.975. Both DBN and Bagging exhibit competitive performance in identifying valid and illegitimate activities in binary form. Bagging achieves a maximum precision of 0.995, recall of 0.997, F1 of 0.996, accuracy of 0.993, and G-mean of 0.982. Consequently, the bulk of fraudulent assignments have been identified and deleted before to their assignment to MCS participants. As a consequence, Bagging results in a 12.19% energy savings for participants and a drastic reduction of affected recruits to 0.25%, compared to a baseline method with no intelligent technique for fake job mitigation that results in a 10.56% effect. In addition, Bagging reduces the proportion of participants affected by bogus tasks from 10.94% to 0.35%.

Thirdly, regarding the FL-backed system for fake task detection in MCS, two FL-based models are implemented with different ML algorithms and different local datasets in participating devices where the aggregated decision leverages risk/loss-awareness with utilities and losses with respect to task values. A traditional centralized platform with ensemble ML algorithms (e.g., Adaboost, Bagging, and XGBoost) for detection is introduced to compare with an FL-based system. In the centralised system, Adaboost and XGBoost have the highest G-mean and accuracy, according to experimental data. In dynamic and

vote-based instances, the FL-based model with a big dataset achieves the same performance as a centralised system, with a G-mean of 0.9743 and an accuracy of 0.9900. With smaller datasets, the federated model obtains 100% overall accuracy in both dynamic and static reputation instances. For dynamic reputation and vote-based federated models, a higher detection performance results in a smaller average loss in terms of task values.

Chapter 4

On Blockchain Integration into MCS via Smart Embedded Devices

Blockchain is a digital ledger technology that enables numerous parties to maintain a decentralised, secure, and tamper-resistant record of transactions and data. This decentralised system eliminates the need for a central authority or middleman to verify transactions [280]. The fundamental concept beside blockchain is that each block of data is sequentially connected to the previous block, forming a chain of blocks. Each block contains a record of multiple transactions which are validated and inserted into the chain by a group of individuals called nodes. Once a block is inserted into the chain, it's unable to be revised or removed, ensuring the immutability of the data on the blockchain. Due to its potential to revolutionise numerous industries, including finance, healthcare, and supply chain, blockchain technology has acquired popularity. It provides transparency, security, and efficacy, allowing for quicker, less expensive transactions with a lower risk of fraud or error [195]. This chapter introduces blockchain-based MCS via embedded devices.

4.1 Blockchain technology in MCS system

Integrating blockchain technology into a traditional MCS platform introduces a new paradigm that adopts the merits of blockchain characteristics such as decentralization, traceability and immutability. Blockchain technology paves the way for the design of a decentralized MCS system. While data is traceable and immutable in a blockchain-based MCS platform, blockchain-based MCS systems overcome various vulnerabilities and improve MCS platform in various ways such as improving worker reliability, introducing a fair evaluation of

the rewarding procedure, preserving sensitive information, and reducing deployment costs [93, 70].

In a nutshell, a blockchain consists of a number of records, named blocks, linking based on cryptographic techniques along with a timestamp and transaction information. It is a type of decentralized and distributed database of records that can be executed and shared between different participating groups. Blockchain is straightforward to deploy as a decentralized system. It can ensure security, primarily because it is distributed and eliminates the need for a central control entity (e.g., a bank in a financial transaction) and, thus, gets rid of the vulnerability resulting from a single point of failure [1]. Furthermore, an immutable ledger is one of the valuable elements of blockchain technology; thus, making an interaction both unalterable and traceable. Therefore, blockchains offer potential opportunities to tackle MCS security and privacy vulnerabilities [63, 36].

A blockchain can be shared publicly between users to create a certifiable transaction record. Moreover, registered users are allowed to add new transactions into a blockchain, which can be reviewed openly at any time; but not changed except in the case of a certain level of authenticity and authorization required to modify data in the chain [5]. As a result, a blockchain contains an unchangeable and complete record within a given network and shares among network sections. A Collective Agreement among system participants needs any update to the blockchain, which cannot be deleted or changed from its original entry. Every transaction is in a public ledger and confirms by a group of participating nodes in the network. Finally, a verifiable and permanent record of every single transaction saves in the blockchain. A blockchain has three fundamental properties:

1) *Decentralization*: A centralized entity is not needed to validate the truth to other users in the system. Participant in a blockchain-based network enables obtaining the previous records of transaction or confirming a new transaction. It means people coordinate individual transactions without relying on a trusted authority [247].

2) *Simplicity*: by adopting a decentralized system, a blockchain eliminates the requirement for and complexity of centralized management. Most importantly, blockchain technology removes the traditional need for third party participation. Third parties lead to potential drawbacks such as cost, risk, information, and control [5]. A blockchain creates a distributed system that achieves data integrity based on consensus without any intervention from a central control unit or node in the network.

3) *Transparency*: Every node in a blockchain has access to the same records spread across an extensive network for all to audit and inspect in real-time, consequently enhancing blockchain transparency. Therefore, the requirement of proof of trust is decreased because the transparency property makes network activities and operations highly visible. As a

result, according to all parties' collective agreement, the system records could be modified and updated by anyone in the network; however, accepting a new block into the blockchain requires meeting certain conditions.

4) *Immutable*: In the context of a blockchain, this stands for the following: once any information has been stored in the blockchain, it cannot be tampered with using the available mechanisms for that (e.g., hash value and proof-of-work) [279]. Section 4.1.2 introduces tamper-proofing of a a blockchain.

4.1.1 Internal structure of a blockchain and smart contract

In addition to a timestamp, a block contains three parts: a transaction data, a hash of the current block, and a hash of the previous block. The hash is calculated when a block is created and forms uniquely identifiable properties. The hash of a block is analogous to fingerprints in that the hash uniquely identifies a block and its associated data. Therefore, modifying the contents of a block changes the hash of the block. Thus, hashes are effective and appropriate tools to detect any modification of data in blocks.

As noted, the hash of the previous block in use is used as a connector to that block. This mechanism is effective in adding blocks into a blockchain and ensure the security of the entire blockchain. Fig. 4.1, demonstrates three blocks connected in a blockchain. As illustrated, every block contains its hash and hash of the previous block. Thus, block $n+2$ points to block $n+1$, and block n+1 points to block n. The $n=0$ block is the only block that cannot point to a previous block. Therefore, block 0 has no previous hash, and the first block is named the genesis block.

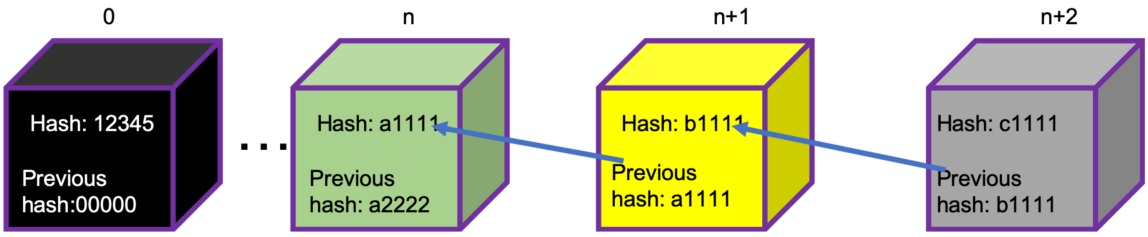


Figure 4.1: Blockchain example

Smart contracts serve for storing contracts in a distributed ledger structure as proposed by Nick Szabofor the first time [178]. A smart contract is a small program designed for

digital verification, facilitation or enforcement of negotiation of service or performance without involving any other trusted party [162]. Due to the elimination of external parties, smart contracts are strong candidates t be used with blockchains [29], and they inherit immutability properties of blockchains alongside their distributed properties. That said, it is impossible to have the contract release the funds or terms because other parties on the network will spot this attempt and immediately invalidate it. In [162], the authors list several smart contracts applications such as supply chain, IoT, healthcare system, insurance and financial system. Smart contracts are programmed to hold all received funds until a specific and predetermined goal is reached. Because smart contracts are stored in a blockchain, all records and transactions are distributed thoroughly.

4.1.2 Tampering prevention of blockchains

If someone is tampering with block $n + 1$, the block hash updates immediately. Consequently, it causes block $n + 2$ and all following connected blocks to become invalid since the hash of the previous block is different in these blocks. It means modifying data in one block will make all consequent blocks invalid. On the other hand, the availability of high-performance computing platforms offers parallel use of multiple processors, often using large numbers of Graphical Processing Units [168]. This last reference also has pointed out that it is possible to exploit advanced data mining capabilities, particularly in cryptographic processes [86], when working with blockchains. That said, the possibility of re-calculating the hash for all subsequent invalid blocks with the intent to re-validate them cannot rule out. Therefore, solely relying on re-evacuating relevant hash codes is not enough to prevent tampering.

The notion of Proof-of-Work (PoW) [121], a consensus mechanism, is introduced to the blockchain to mitigate the re-evaluate problem. PoW is a preliminary condition for a block to be accepted by network participants. Miners, who undertake the computing tasks in reaching consensus, should complete a PoW for a block regarding access to all data in the block. The difficulty with PoW is adjusted to limit the rate of generating new blocks. PoW is a procedure to validate the inbound and outbound data against security attacks such as Denial of Service (DoS) or spamming. The essential enabling factor of spam or DoS attacks is low cost for malicious users. PoW comprises a set of proposals (e.g., upfront payment prior to messages, requirement for resource-intensive computation, and numerous memory operations) to boost both computation and monetary costs[141]. The study in [141] demonstrates that PoW can be integrated with per-email spam mitigation and source reputation to mitigate spamming mails more effectively than the traditional anti-spam system. In the scenario of combining PoW with stepwise reputation mechanism,

the system achieves 99% detection accuracy and 1% of false positives. Furthermore, in applying cryptocurrencies (e.g., bitcoin), on average, 10 minutes of processing time is needed to compute the required PoW and insert a new block into the blockchain [53]. This time-consuming approach makes it challenging to tamper with the blocks as it would be necessary to re-calculate the PoW for all subsequent blocks.

Thus, the security and robustness of the blockchain build on hash and the PoW mechanisms across blocks. Besides, blockchains also can be secured by deploying a peer-to-peer network that allows new blocks to join it with the verification process. For instance, as illustrated in Fig. 4.2, a new block $n+3$ is to be added into the blockchain that contains $n+2$ blocks. The new block is verified by all blockchain users to assure that it has not been altered or tampered. Once the block passes the verification, it is added by all users into their blockchain.

Nodes in the same network reach consensus by agreeing on the validity of the blocks. As a result, in order to tamper with a blockchain, an adversary needs to: 1) tamper all blocks on the same chain, 2) re-compute a PoW for all blocks, and 3) reach an agreement with more than 50% of the miners to rewrite the ledger, in order to reach consensus. If the above three conditions hold, every node will accept the tampered block. However, the probability of these three conditions lining up is considered negligible showing in tutorial video in [57]; but not impossible in non-deterministic polynomial-complete time given massive computing resources coupled with collusion among a large percentage of miners. This video [57] introduces blockchain technology briefly, including blockchain working procedure and solving problems. In [57], the difficulty of tempering a blockchain is illustrated due to PoW and consensus techniques.

4.1.3 Blockchain applications

Blockchains can be used in various areas. For example, based on attribute signcryption, a blockchain-based clouding data sharing environment is designed to secure the data and boost the efficiency [68]. The study in [10] demonstrates a blockchain-based certificateless public-key signature system that uses bilinear pairing with the aim of supporting conditional privacy-ensuring authentication during the message exchanges between vehicles and infrastructures. This designed scheme minimizes the computation cost of generating signatures and verifying signatures and speeds up the verification process via batch signature verification and aggregated verification. Blockchain technology is deployed in WBANs using certificateless authenticated vital agreement [172]. In [102], the authors present an UTXO-based blockchain that results in 2.9 to 4.5 times memory storage reduction. The study in [243] introduces a blockchain implementing using Hyperledger Fabric in

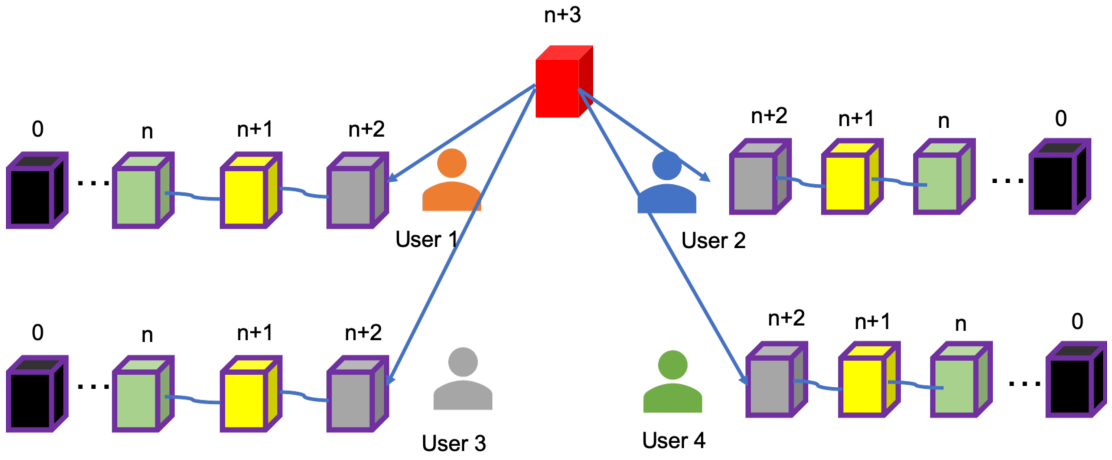


Figure 4.2: Example of a new block adding into blockchain with four users

a crowdsourced energy system, where crowdsourcing is leveraged in the generation of small-scale energy, as well as energy trading. It is worth to note that energy is crowdsourced from sources like distributed energy providers or electric vehicles. Furthermore, financial transactions can be remodelled using blockchain technology. The study in [235] outlines an implementation of blockchain using Ethereum, a decentralized software platform that enables smart contracts and distributed applications, one of which is a cryptocurrency. Moreover, cryptocurrencies support funding and monetary system features. People can use their cryptocurrency to trade with others for business purposes [232].

A benefit of blockchains can be found in the following bitcoin use case. A sender transfers funds to a geographically located recipient in a foreign jurisdiction. Under the traditional business model, the fund transfer requires a third party for the transfer. Removing this third party layer by replacing it with blockchain technology reduces transaction costs and time. Thus, blockchain-based cryptocurrencies are beneficial compared to traditional transaction models because they are distributed, transparent, timely and eliminate overhead. In addition to cryptocurrencies, electronic voting and recommendation systems can utilize blockchains [120, 180, 20]. The authors in [180] investigate the loopholes of the existing voting systems and the idea of a blockchain-integrated electronic voting system. Meanwhile, they demonstrate challenges and obstructions when deploying blockchains in electronic voting systems. The study in [120] discusses a blockchain-based approach to alleviate two critical weaknesses (e.g., voter fraud and voter access) in current voting

systems. With blockchain-enabled voting and recommendation systems, voter fraud is decreased, and voter access is raised. Thus, eligible voters who have allocated tamper-proof identities and encryption keys could vote anonymously through their electronic devices. Consequently, blockchains create tamper-proof audit trails, which can be integrated with e-voting and recommendation systems [120]. Moreover, as reported by the study in [20], an innovative e-voting system is implemented by leveraging open-source blockchains. The blockchain-based system can be utilized for both recommendation systems and elections considering particular standards. The proposed system is safe, reliable, and anonymous, contributing to raising on-line voting and increasing the number of voters who trust an e-voting system. In [163], blockchain-based recommendation systems are introduced which satisfy the requirement of various parties (e.g., government and companies) in terms of the security level applied to specific information.

Blockchain can address various vulnerabilities, for instance, using smart contracts to grant authority on access to IoT data. A survey paper [224] focuses on security issues in IoT systems and lists blockchains-based solutions to IoT systems including 1)data trustworthiness [139], 2) information privacy [11], 3) fault tolerance [179], 4) elimination of third party threats [22], 5) access management [187]. The study [78] demonstrates DoS attacks mitigation in IoT system using blockchain techniques. Besides, some studies tackle the decentralized data ownership issues in IoT [138]. Collection, storage, and validation are the building blocks of a provenance system, and the introduction of a blockchain network ensures tamper-resistance and accountability of the tasks across computing resources [12]. Furthermore, high volumes of IoT data saving and preservation can be made possible via blockchain-based solutions [131]. Finally, the authors in [229] demonstrate a methodology for designing IoT-cloud environments using blockchain. Blockchain technology introduces opportunities to deploy a decentralized MCS platform quickly that would address several vulnerabilities (e.g., personal information leakage in centralized server) in a centralized MCS platform.

4.2 Practical Byzantine Fault Tolerance-based Robustness for MCS

Based on the analysis of the benefits of blockchains listed earlier and perspective blockchain-based applications, this thesis introduces blockchain integration into an MCS system, in order to build a decentralzied MCS platform instead of a conventional centralized one. A sensing task contains sensitive information (e.g., distribution location, distribution time,

recruitment requirements, and awards) that could be modified and tampered with by attackers. The malicious users aim to impact the recruitment process and data evaluation procedure to improve the probability of being selected and getting more rewards. It is critical to store sensing tasks scrupulously to avoid task data tampering. However, most of the related works do not discuss tampering with the sensing in MCS systems. To the best of our knowledge, this work presents the first study to introduce the integration of blockchain technology into an MCS system to prevent sensing task data from being tampered with. Furthermore, based on the literature, blockchain offers a robust solution to overcome the vulnerabilities of a centralized system such as the single point of failure and potential data breach. A common issue with MCS is that an MCS system is generally designed and built in a centralized manner. With the motivation to address the issues with the centralized MCS system, a blockchain-enabled MCS framework is designed to build a decentralized protection framework for an MCS system. As a consensus algorithm is the core entity of a blockchain, this work selects the Practical Byzantine Fault Tolerance (PBFT) consensus in blockchain because of its lightweight nature and suitability for resource-limited environments such as mobile phones. Lastly, to raise fake task detection accuracy, an ensemble learning approach is implemented in the blockchain network that inherits the benefits of multiple ML algorithms (i.e., base estimators) for precise decisions.

4.2.1 Thread model and the proposed system overview

There are three entities in a conventional MCS framework: requesters, a centralized MCS platform, and participants. An MCS activity begins with a requester initiating sensing tasks that are submitted to the MCS platform. In the next step, the MCS platform collects tasks (task assemble and storage) and distributes to participants for recruitment (i.e., task-participant matching). The Data Flow Diagram (DFD) in Fig.4.3 illustrates the data flow for task submission, task collection and distribution. Meanwhile, trust boundaries are illustrated to demonstrate the elements that are confronted by security threats. Threat modeling is a structured way of analyzing and solving problems to identify, quantify, and respond to threats, using abstract approaches to anticipate potential risks. Fig. 4.4 presents threat identification via $STRIDE$ thread model. $STRIDE$ threat modeling is a threat modeling method proposed by Microsoft to classify the threats under Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service (DoS), and Elevation of data Privilege [35]. $STRIDE$ threat model can almost cover most of the current security problems. Traditional MCS systems are vulnerable to various threats. When $STRIDE$ threat model is applied to analyze each element in DFD, the threats that are identified can be summarized as shown in Fig. 4.4. Just to mention one, participants may temper

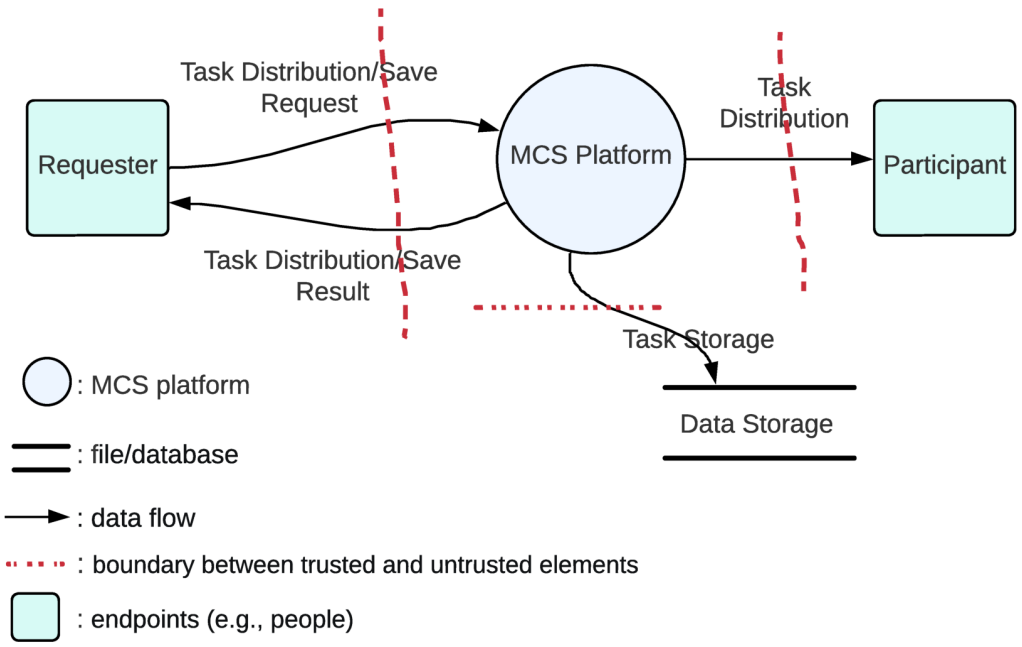


Figure 4.3: DFD of MCS system

and spoof collected data in order to maximize rewards. It is almost impossible to have a single solution against all these threats in an MCS system; hence, this work focuses on data tampering and DoS attacks via flood of fake tasks in MCS platforms and data storage (with a red tick in Fig. 4.4).

An innovative blockchain-enabled MCS framework is proposed to address data tempering and DoS threads as shown in Fig. 4.6. The proposed system consists of three main components: requesters (e.g., malicious requester and regular requester), an a decentralized MCS platform, and participants. Requesters and participants show the same responsibilities as in a conventional MCS framework. Fig. 4.5 illustrates data flow of the proposed system. In the proposed system in Fig. 4.6, the malicious requester is malicious entity in the system, who attacks MCS system via submitting fake tasks. A fake task drains more energy from participants' devices and clogs the MCS platform to prevent the MCS platform from providing services to normal users, which is one of the most critical threats in the MCS systems [272]. Fake task attack will clog MCS platform to make it out-of-service for normal requesters so fake task attack is a kind of DoS attack.

Element	S	T	R	I	D	E
Requester	✓	✓		✓		
Participant	✓	✓				
MCS Platform	✓	✓	✓	✓	✓	✓
Data Storage	✓	✓		✓	✓	
Data Flow		✓		✓		

Figure 4.4: *STRIDE* thread model for MCS system. This work focuses on threads with red ✓.

The role of the proposed MCS platform provides functions to collect submitted tasks from the requester and distribute tasks to participants, which is the same as a conventional MCS platform. Furthermore, it also contains a function to detect fake tasks via an ensemble learning approach and save tasks into a blockchain network. On one hand, the ensemble model prevents fake task to clog MCS platform so it is an effective approach to avoid DoS attack. On the other hand, blockchain technique is born to prevent from sensing task tampering. Sensing tasks, regardless of their legitimacy, are transactions of the blockchain-based decentralized MCS platform. Data of fake tasks look similar to legitimate tasks which does not bring security issues to blockchain until saving them in the MCS platform or distributing them across participants. It means that blockchain taking either fake or legitimate task data to calculate cryptographic hash will pass verification. The blockchain-based MCS platform replaces a centralized MCS platform to communicate with requesters, which prevents the single point of failure issue in a centralized MCS platform. In the

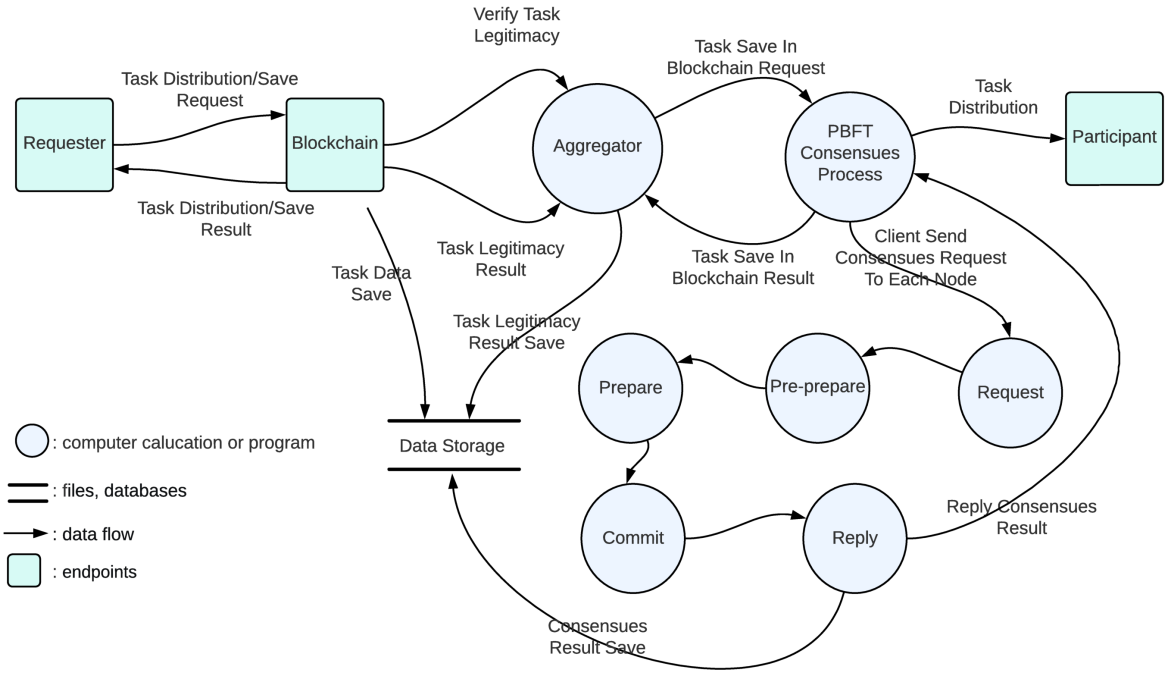


Figure 4.5: DFD of the proposed system

designed MCS platform, there is no single party that has authority to control all such as collecting sensing tasks or distributing task to participants. Meanwhile, there is no single party that makes arbitrary changes to the blockchain's data or rules. Decisions are typically made through consensus by involving network participants. This makes the MCS platform decentralized. As shown in Fig. 4.6; if a task is estimated as legitimate, the blockchain provokes the PBFT protocol to determine whether the transaction/task can be added to the blockchain. Meanwhile, the predicted legitimate tasks are distributed to participants for recruitment and executing the subsequent procedures in MCS. On the other hand, a task predicted as a fake is dropped, and the PBFT processes are not triggered. Since PBFT tolerates inactive nodes to reach consensus, the proposed system is robust enough to resist node faults.

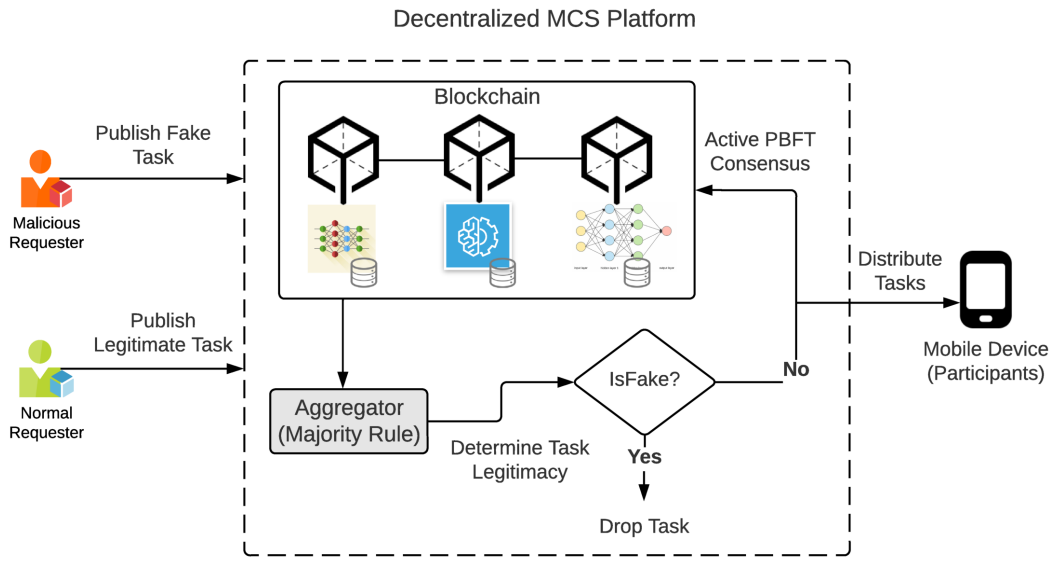


Figure 4.6: The proposed blockchain-based MCS system

4.2.2 Problem definition

This work pays attention on three issues: 1) to design a fake task detection method applicable for a decentralized MCS system in order to protect participants and MCS server, 2) to save legitimate sensing tasks into a blockchain network to avoid task data tampering, 3) to tolerate faulty devices with motivation to build a robust and sturdy MCS system. For defining the problem more precisely, we firstly introduce the parameters required for problem formulation.

In this work, an ensemble learning approach is designed to estimate the legitimacy of tasks. Prediction accuracy is a significant metric to evaluate the detection performance. The number of fake tasks and legitimate tasks is denoted as T_f and T_l, respectively. A task may be estimated as a legitimate one that can be either accurate prediction or false prediction, which necessitates the analysis of the number of fake and legitimate tasks. Parameters w and v stand for the number of legitimate tasks under correct prediction and the number of fake tasks under incorrect prediction, respectively.

To save sensing tasks into the blockchain, two evaluation metrics, namely Ratio of Fake Tasks ($RoFT$) and Ratio of Legitimate Tasks ($RoLT$), are used to evaluate blockchain's ability to save legitimate and fake tasks. $RoFT$ is the number of fake tasks saved successfully over total fake tasks. $RoLT$ is the number of legitimate tasks saved successfully over

total legitimate tasks. The objection of blockchain to store task data is to protect legitimate task data from tampering rather than fake tasks. Therefore, Our purpose is to ensure that the minimum fake and the maximum legitimate tasks are stored. Consequently, the smaller $RoFT$ and the higher $RoLT$ are better performance indicators for the proposed system. As presented in Fig.4.6, a task predicted as a legitimate one would be sent to the blockchain network, triggering consensus procedures.

Meanwhile, a task would be stored in the blockchain if it obtains an agreement (consensus) from the blockchain. Blockchain consensus procedures rely on the number of workable blocks so N, k, and t are used to represent the total number of nodes, the number of faulty nodes, and the number of workable nodes in the chain, respectively.

With a motivation to ensure MCS system robustness, PBFT consensus is applied, which tolerates faulty nodes to reach an agreement in a blockchain. Individual nodes could be inactive (e.g., unstable network during consensus procedures), namely faulty nodes. PBFT resists several faulty nodes if the number of faulty nodes is less than 1/3 total nodes. Therefore, k should be less than 1/3 to reach a consensus under PBFT. Meanwhile, fault probability (p) for each node in a blockchain is considered. The probability that the blockchain reaches an agreement for a task is denoted by the parameter p_{ft}. In order to compare the proposed decentralized framework, the probability of a task saved successfully in a centralized MCS system and non-fault tolerant (NFT) (without PBFT) MCS systems are presented, represented by p_c and p_{nf}, respectively. $RoLT$ / $RoFT$ is compared in three scenarios as following:

- Under different task detection methods among ensemble learning and base ML algorithms (e.g., SVM, Adaboost, KNN, Decision Tree)
- Under different system architectures such as centralized and decentralized systems
- Under PBFT-based Fault tolerance systems and NFT systems

Table 4.1 lists symbols and notation commonly used in the rest of Section 4.2.

Proposed MCS Framework

A decentralized MCS system is proposed integrating blockchain technique, which relies on a PBFT protocol to achieve consensus for a single data value (a task submitted by requester) among the distributed nodes in the blockchain. It means a task can be saved in the MCS platform determined by the PBFT procedures. Indeed, PBFT consensus

Table 4.1: Symbols and Notations

p	Fault probability for each node in the blockchain
N	Total number of nodes in the blockchain
k	Number of faulty nodes tolerated by PBFT to reach consensus
t	Number of active nodes in blockchain
p_{ft}	Probability for achieving consensus under PBFT
p_{nf}	Probability for achieving consensus without fault tolerance
p_c	Probability for a centralized system deciding a transaction if saving
w	Number of legitimate tasks with correct prediction
v	Number of fake tasks with incorrect prediction
T_f	Number of fake tasks in test dataset
T_l	Number of legitimate tasks in test dataset
$RoFT$	Ratio of Fake Tasks
$RoLT$	Ratio of Legitimate Tasks
NFT System	Non-fault tolerant MCS system
CL System	Centralized (CL) MCS platform

needs time for calculation to achieve agreement; therefore, prompt elimination of fake task submissions is not expected from the system. However, when MCS tasks are handled in a participatory manner, task-participant matching is not necessarily done immediately. More so, task-participant matching involves some computational overhead to determine rewards and reach an equilibrium in terms of the (reward) utilities of the participants and utility of the MCS platform. In other words, a sensing task seeks and reserves the resources of an MCS system in a future time slot. Therefore, it can be anticipated that PBFT can be applied between the task submission time and participant-task matching in parallel to the game/incentive theory-based optimization models empowered to select the participants. The proposed framework is designed to pledge for fake tasks detection using the ensemble learning algorithm in the blockchain. The voting-based ensemble learning approach is proposed for preventing malicious sensing tasks. In this work, four blocks are selected randomly for arranging Adaboost, KNN, Decision Tree (DT), and SVM. The study in [99] demonstrates energy consumption using a smart controller representing an edge device in an IoT system that deploys a Random Neural Network (RNN). A Random NN algorithm to estimate the occupancy and project mean vote-based setpoints for heater, conditioning, and HVAC management is embedded into the smart controller. The author demonstrate 30.16 KWh in the smart controller with the Random NN. In this work, the types of the nodes in the blockchain are not specified so it is difficult to obtain an accurate energy consumption.

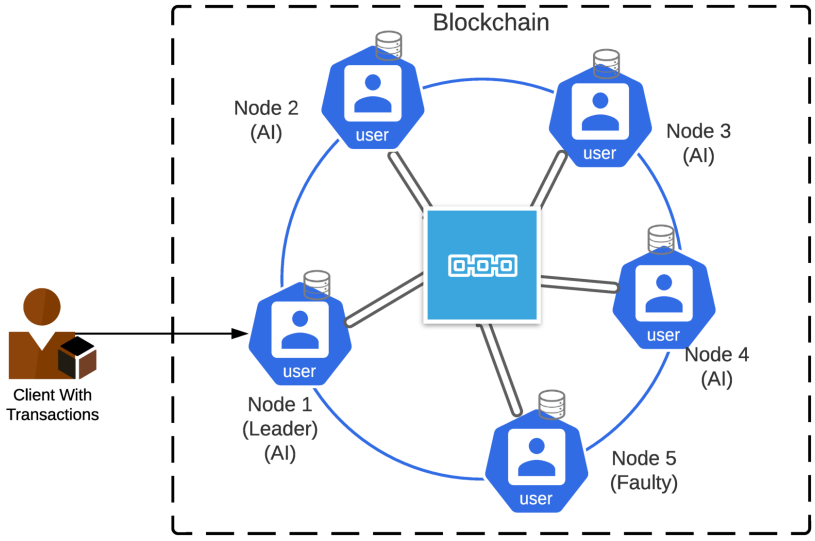
Nodes in the blockchain network are similar to devices in the referred system, that are battery-powered edge equipment. Therefore, it can be expected that energy consumption of the devices in blockchain network could be close to 30.16 KWh even when deploying the machine learning algorithms. An Aggregator collects individual ML model prediction results and makes a final decision based on the majority rule. If the Aggregator predicts a task as legitimate, PBFT consensus processes are triggered in blockchain to decide whether this task can be added to the blockchain.

PBFT-based Blockchain-Driven MCS

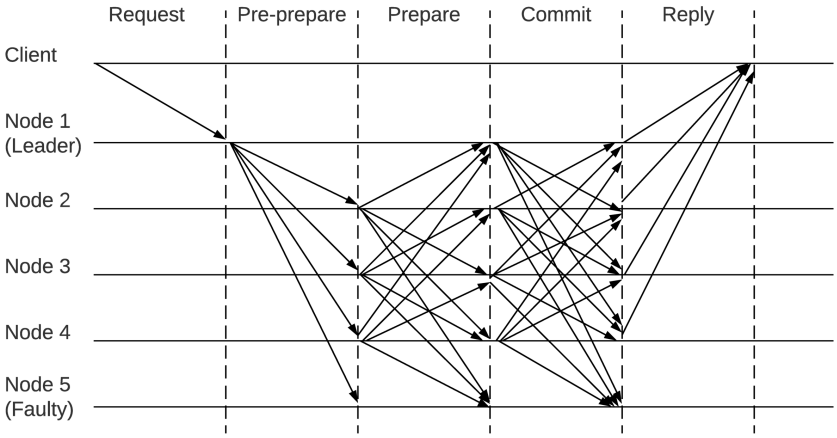
PBFT is selected as the consensus protocol in the proposed blockchain. PBFT addresses a fundamental problem that computing nodes in a distributed network agree on a decision on whether some of the nodes fail or act dishonestly [135]. BFT is a property of a system rather than an algorithm itself. PBFT reaches a consensus for transactions if less than 1/3 nodes in the blockchain are inactive/failed. Fig. 4.7a shows an example blockchain with 5 nodes. $Node1$ is a leader/primary node in this chain, the only node communicating with clients. The primary node is selected randomly or taking turns among all the nodes. This means that the client communicating with all nodes in the blockchain shares the same probability rather than communicating with one node. If $Node1$ does not work, a new leader node will be selected randomly among the remaining 4 nodes and continues communicating with the client in this example. Therefore, the failure of one node does not affect the operation of the entire system. A client does not communicate with a fixed entity. All entities can handle the requester's tasks uniformly in a decentralized MCS platform whereby the single point of failure is addressed. Fig. 4.7a shows an example with one client, representing a MCS requester. Indeed, there should be various requesters communicating with the blockchain who have sensing tasks to be added in the blockchain. In a general manner, clients in a blockchain are the people who have new transactions waiting to be added to the blockchain. In this work, clients are requesters (either malicious or regular requesters), and transactions are sensing tasks.

In a PBFT based blockchain, the calculation and decision-making processes are performed by a group of nodes that participate in the consensus protocol. It differs from PoW or PoS blockchain. PBFT based blockchains do not have miners. Instead, PBFT operates on a different consensus mechanism that relies on a set of replica nodes to reach consensus. PBFT protocol consists of 5 processes for reaching consensus as shown below [134]:

- **Request**: Client/Requester sends a new transaction/sending tasks to the *Leader* node, which is an initial step.



(a) Clients communicates with blockchain.



(b) PBFT working procedure (reproduced from [200])

Figure 4.7: An example of a 5-nodes (1 faulty node and 4 normal nodes) blockchain and PBFT working procedure

- **Pre-prepare**: A *Leader* node is selected randomly, who sends $Pre-prepare$ message to other nodes. A node accepts the $Pre-prepare$ message so long as it is valid. A $Pre-prepare$ message contains a sequence number, signature, and other metadata that allows nodes to determine message validity.
- **Prepare**: If a node accepts the $Pre-prepare$ message, it sends out a $Prepare$ message to everyone else. $Prepare$ messages are accepted by receiving nodes after verifying their validity with respect to the sequence number, signature, and other metadata. A node will be considered as $Prepared$ if it has seen the original request from the leader node, pre-prepared, and seen no less than $2k$ $Prepare$ messages that match its $Pre-prepare$ message. Here, k stands for the number of Byzantine faults.
- **Commit**: After the nodes are prepared, they send out a $Commit$ message. If a node receives $2k$ $Commit$ message, they can carry out the Client request.
- **Reply**: The Client waits for $2k+1$ of the same *Reply* message. Since the maximum number of Byzantine faulty nodes is k, the number of received *Reply* messages needs to be at least $2k+1$. This ensures that the response is valid.

Fig. 4.7b demonstrates a consensus example in a 5-nodes blockchain. In this example, $FaultyNode$ drops out due to lose network connection, and it cannot reply to any message in the following steps. In this work, we define a faulty node as the one which can not reply to messages during PBFT consensus procedures. Thus, the faulty nodes are excluded from PBFT consensus automatically. However, the *Leader* node does not know that yet, who still sends $Pre-prepare$ message to the inactive node. In this example, the number of Byzantine faulty nodes k is 1, which is less than 1/3 of faulty node (1.6). Meanwhile, 1 is the maximum faulty node for PBFT to make consensus under total 5 nodes chain. Therefore, the example of a 5-nodes blockchain can theoretically achieve consensus with 1 Byzantine faulty node.

Consensus in the PBFT-based MCS Framework

The number of faulty nodes is k, which needs to be less than one-third of total nodes N for PBFT to reach an agreement.

$$\frac{k}{N} < \frac{1}{3} \tag{4.1}$$

Maximum value of k satisfying (4.1) can be defined as

$$k_{max} \triangleq \lceil \frac{N}{3} - 1 \rceil \tag{4.2}$$

where the ceiling function $\lceil \cdot \rceil$ maps x to the smallest integer greater than or equal to x.

Total number of nodes N can be written as the sum of the number of faulty nodes k and the number of active nodes t as

$$N = k + t. \tag{4.3}$$

Obviously, the number of active nodes sets the relation between the total number N and the number of faulty nodes as

$$t = N - k. \tag{4.4}$$

where t denotes the number of active nodes in a blockchain network, that are involved in validating a transaction in the PBFT. Meanwhile, from (4.1) and (4.3), a relation between the number of faulty nodes and the number of active nodes is obtained as

$$t \geq 2k + 1, \tag{4.5}$$

It means that PBFT reaches agreement should satisfy the number of workable nodes greater than two times of the number of faulty nodes.

Provided that each node has a fault probability p, under constraints $0 \leq p \leq 1$, a node has workable status probability $(1 - p)$. The PBFT protocol achieves consensus with a probability p_{ft} under the constraint (4.5). A PBFT can reach an agreement with 0 faulty node, 1 faulty node, and up to k_{max} faulty nodes.

If the number of faulty nodes is i, it says i node(s) is/are inactive and other nodes $(N - i)$ work well. The i faulty nodes are selected randomly among all nodes so the number of combination of N items taken i at a time needs to be considered as formulated as

$$C\{N, i\} = \frac{N!}{(N - i)!i!} \tag{4.6}$$

Thus, a system probability with i faulty nodes is generalized as

$$p_{ft}^{i} = C\{N, i\} p^{i} (1 - p)^{(N-i)}. \tag{4.7}$$

If there is 0 faulty node, it means all nodes in workable status. In this case, Equation (4.7) turns out to be

$$p_{ft}^{0} = (1-p)^{N} \tag{4.8}$$

As a PBFT-based system tolerating k_{max} faulty nodes, it means that the system where PBFT reaching an agreement consists of cases of system with all workable nodes, 1 faulty node, 2 faulty nodes, and up to k_{max} faulty nodes. Probability of a PBFT-based system to make consensus successfully with k faulty nodes is the sum of probabilities of each faulty node case up to k faulty nodes.

$$p_{ft}^{k} = \sum_{i=0}^{k} p_{ft}^{i} \tag{4.9}$$

From (4.7), (4.9) turns out to be

$$p_{ft} = \sum_{i=0}^{k} C\{N,i\} p^{i} (1-p)^{(N-i)} \tag{4.10}$$

(4.10) represents consensus probability of a PBFT system that depends on node faulty probability p, total number of nodes N and the number of tolerated faulty nodes k. With particular p and N, the maximum probability of a system reaching an agreement under PBFT is determined. As p is restricted between 0 and 1, p_{ft}^{i} should not be a negative value. Thus, p_{ft} results in peak value when k is the maximum one with specific p and N. Therefore, (4.10) is re-formulated as

$$p_{ft}^{max} = \sum_{i=0}^{k_{max}} C\{N,i\} p^{i} (1-p)^{(N-i)} \tag{4.11}$$

We always consider the maximum probability of a PBFT-based blockchain to reach an agreement so to simplify it, expression p_{ft} is used to instead of p_{ft}^{max} under constraints of k the same as k_{max}.

In order to compare with the proposed system, NFT and centralized systems are considered and formulated. An N-nodes decentralized system without fault tolerance (NFT) results in system working probability p_{nf} as shown in (4.12). It means every node should be valid with probability $(1-p)$. Thus, a N-nodes NFT system contributes the probability N power of $1-p$.

$$p_{nf} = (1-p)^{N}, \tag{4.12}$$

where demonstrates a NFT system workable probability depends on system scale (N) and individual node fault probability (p).

As to a centralized system, it contains one node to decide a task is being stored or abandoned which is the same as a conventional centralized MCS system. A traditional MCS platform is a center entity to communicate with requester and participants. If the centralized MCS platform does not work, the entire MCS system stops working. Therefore, working probability of a MCS framework is the same as working probability of the node $(1-p)$.

$$p_c = 1 - p \tag{4.13}$$

Therefore, a task storage probability p_s defined in (4.14) depends on system architecture (e.g., centralized, PBFT-based blockchain, NFT system).

$$p_s = \begin{cases} \sum_{i=0}^{k} C\{N, i\} p^i (1-p)^{(N-i)}, & \text{PBFT} \\ (1-p)^N, & \text{NFT} \\ 1 - p, & \text{Centralized} \end{cases} \tag{4.14}$$

As to task saving to blockchain network, it rely on probability of a system to allow saving a task (p_s) and prediction prediction results. As introduced in system architecture Fig. 4.6, the tasks predicted as legitimate are send to blockchain network and trigger PBFT consensus procedures. Legitimate tasks could be predicted correctly by ensemble learning-based approach, that are transferred to blockchain. Meanwhile, fake tasks estimated as legitimate will be sent to blockchain as well, which is side effect by ensemble learning model due to non-100% prediction accuracy. The proposed framework aims to save more legitimate tasks and less fake ones. $RoLT$ and $RoFT$ are utilized for evaluating system performance of task saving, which are formulated in (4.15) and (4.16), respectively.

$$RoLT \triangleq p_s \frac{w}{T_l}, \tag{4.15}$$

where w is the number of legitimate tasks predicted accurately and T_l is the number of legitimate tasks in test dataset.

$$RoFT \triangleq p_s \frac{v}{T_f}, \tag{4.16}$$

where v is the number of fake tasks predicted as legitimate ones and T_f is the number of fake tasks in test dataset. (4.15) and (4.16) represent $RoLT$ and RoF depending on task detection performance and system architectures.

Applying Ensemble Learning into PBFT-driven Blockchain-enabled MCS System

Although various studies have demonstrated ML-based methods for fake task identification, the detection performance could be improved further. Ensemble learning models have proven to result in more promising performance than individual ML algorithms [210]. In this work, an ensemble learning is designed and combined with the PBFT-based blockchain which aims to identify fake tasks accurately. The ensemble learning method consists of four different ML algorithms (e.g., Adaboost, KNN, DT, SVM), deploying in four blocks in the blockchain. Each block have the same training dataset to train individual ML models and make predictions independently. Meanwhile, an ensemble learning approach comprises a centralized server entity, namely the aggregator. The aggregator collects base-estimators prediction results and makes a final decision. Therefore, the centralized aggregator still has single-failure issue. It means if the entity does not work, the introduced system can not identify fake tasks. The designed ensemble learning approach in the decentralized MCS platform is illustrated in Fig. 4.6. A block is considered a decentralized device that contains a base classifier. Meanwhile, individual ML estimation results are uploaded to a server which is the Aggregator module in Fig. 4.6. Based on the majority voting rule, the final decision is decided by the Aggregator. According to Fig. 4.6, the Aggregator decision is sent back to decide if PBFT consensus executed or not. If the Aggregator estimates the task as fake one, system drops it; otherwise, the PBFT is triggered to determine if this task can be saved in the blockchain.

Implementation of Blockchain with PBFT protocol

In this work, Python version 3.9 is used with some useful libraries such as *aiohttp* and *hashlib* to implement PBFT in a blockchain. In implementation of the PBFT protocol, all messages including $Request$, $Pre-prepares$, $Prepare$, $Commit$, and $Reply$, contains Client request data and other data for message verifying as introduced in Section 4.2.2. Therefore, Client transactions/tasks are distributed to all nodes after $Pre-prepare$ message. ML algorithms utilize training data but training does not require the entire consensus procedure in the blockchain. Therefore, training data assembles during $Request$ and $Pre-prepare$ steps rather than entire PBFT processes as illustrated. in Fig. 4.8.

Tasks in the test dataset go from $Request$ step to $Reply$ step in PBFT consensus protocol to reach consensus in the blockchain as shown in Fig. 4.7b. Fig. 4.9 illustrates a fragment of the blockchain we implemented, including the genesis block and the first block with sensing task data in it. A genesis block is an initial block of a blockchain that

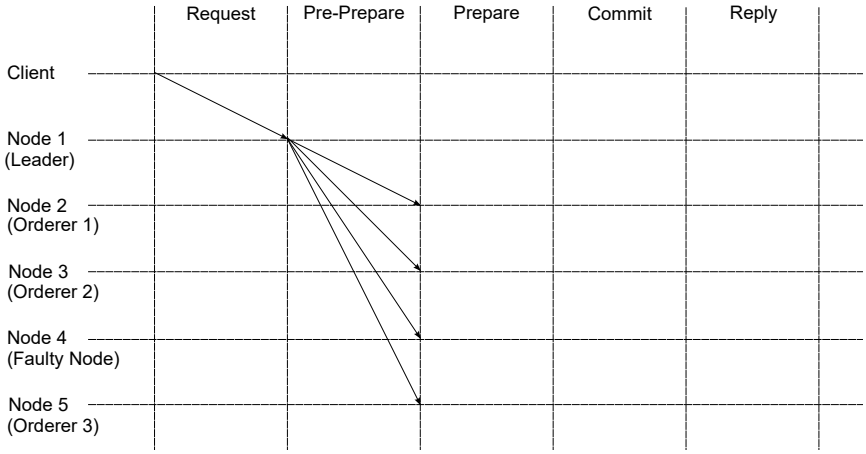


Figure 4.8: Training dataset accumulation in individual nodes in blockchain after Request and Pre-prepare messages of PBFT

typically does not contain particular data. Blocks are linked together via cryptography. Block 1 is the one after the genesis block and links the genesis block, so the previous hash of block 1 is the same as the hash value of the genesis block. Each block consists of five parts as follows.

- **Hash**: Cryptographic hash is calculated according to data saved in this block.
- **Index**: It stands for a unique block index in the blockchain.
- **Previous hash**: Block links to the block which contains hash value the same as this value.
- **Timestamp**: Block generates time.
- **Transactions**: It stands for data saved in this block. In this work, sensing tasks save in blocks. Meanwhile, client index is saved since more than one clients/requesters are allowed to send tasks to blockchain. Blockchain also assigns a unique index for each transaction for easy tracing and separating.

From Fig. 4.9, a block (except the genesis block) saves 5 tasks with a *client* ID and an *unique* ID for every MCS task. For instance, $[0, 2]$ stands for Client 0 submitting a task assigned with a unique ID 2 by blockchain. It means that task 2 data is a part of block 1 and verified by PBFT consensus protocol in the blockchain. Meanwhile, both fake tasks

```
    "hash": "859a66ab9c0d5c415b1f1cab7db8554c8b5de27bb3a7f91bea4fa4fd22eaedc3",
    "index": 1,
    "previous_hash": "724d22633c6de11ea385753a340a7e13996064251fea63252b8e78658ce6ab23",
    "timestamp": "Mon Apr 11 21:48:20 2022",
    "transactions": [
        [
            "[0, 0]",
            "MCS tasks ['ID,Latitude,Longitude,Day,Hour,Minute,Duration,RemainingTime,Resources,Coverage,OnPeakHours,GridNumber,Ligitimacy']"
        ],
        [
            "[0, 1]",
            "MCS tasks ['14,48.36682189,-81.43076265,5,13,13,60,60,10,100,FALSE,151699,1']"
        ],
        [
            "[0, 2]",
            "MCS tasks ['20,48.53052149,-81.39285937,5,3,12,40,40,2,100,FALSE,436073,1']"
        ],
        [
            "[0, 3]",
            "MCS tasks ['22,48.47408561,-81.32675138,5,7,30,10,10,7,100,TRUE,341291,0']"
        ],
        [
            "[0, 4]",
            "MCS tasks ['36,48.60305844,-80.88628052,5,8,44,50,50,4,100,TRUE,568838,1']"
        ]
    ]
{
    "hash": "724d22633c6de11ea385753a340a7e13996064251fea63252b8e78658ce6ab23",
    "index": 0,
    "previous_hash": "0",
    "timestamp": 0,
    "transactions": [
        "Genesis Block"
    ]
```


Figure 4.9: Fragment of PBFT-based blockchain for task storage under Timmins dataset

Table 4.2: Distribution of training and test datasets. Leg: Legitimate

		Training	Test			Training	Test
Timmins	Fake	82	15	CR	Fake	1572	101
	Leg	750	153		Leg	10422	2389
	Total	**832**	**168**		**Total**	**11994**	**2490**

and legitimate tasks are saved in the blockchain. For example, Fig. 4.9 demonstrates that the transaction in $[0, 3]$ saves a fake task with last field 0 whereas the last transaction in $[0, 4]$ stores a legitimate task with last field 1. Blockchain aims to resist data tampering; hence, it is difficult to determine task legitimacy during the cryptographic hash procedure. Therefore, both fake task and legitimate task can be saved successfully in the blockchain, when ensuring task data consistency and intact. In this work, we consider all tasks are original and intact without tampering.

4.2.3 Performance evaluation

This section presents performance evaluation under Timmins and CR datasets. Initially, starting with measuring a system probability of saving a task, it considers various system

architecture, node fault probability, and system scales. Then, ensemble learning-based prediction results are presented, and fake task saving results $RoFT$ and legitimate task saving results $RoLT$ are demonstrated.

Blockchain network performance

Blockchain network performance is critical to evaluate the proposed framework. In this subsection, we evaluate the blockchain state in terms of average block size, blockchain size, total blockchain creation time, individual block time, hash rate, and the number of blocks committed in 30 minutes. To illustrate a specific real world example, the test in this work is executed in MacBook Pro (13-inch, 2020, Two Thunderbolt 3 ports), with the following hardware configuration: processor (1.4 GHz Quad-Core Intel Core i5), memory (8 GB 2133 MHz LPDDR3), and graphics (Intel Iris Plus Graphics 645 1536 MB), that indicates the rated power as 60.9W (voltage 20.3V and current 3A). Thus, blockchain network performance depends on the test environment. We take a 5-node blockchain as an example to present the test results and consider latency to represent transmission delays. Transmission delays are commonplace in practise that may be caused by a number of factors relating to the characteristics of the equipment and network configuration, as well as network disruptions. In this study, latency is assumed to follow uniform distribution in range 0 to 1 second. It means a node needs to wait for the delay time (latency) after receiving PBFT messages prior to processing the obtained message. When evaluating the performance of a blockchain network, it is assumed that all tasks from the test dataset are sent simultaneously. This assumption is rooted in the scenario where multiple requesters, both sending fake and legitimate tasks, submit their requests to the decentralized MCS platform concurrently. The intention behind this is to streamline the calculation of blockchain network performance, eliminating the need for waiting periods for incoming transactions. Regarding the number of generated blocks in 30 minutes, the criteria is justified on the basis of the study in [158]. As for the blockchain size, we calculate total bytes in the chain. Meanwhile, average individual block size is the total size of the blockchain divided by the number of blocks. Total time stands for the time elapsed while the blockchain generates all blocks. Therefore, single block time is the total time divided by the number of blocks. A hash rate is the number of calculations that can be completed per second. In this work, $SHA-256$ algorithm [157] is used to calculate the hash. Hash rate is calculated as the total bytes saved in a block divided by the execution delay of the SHA-256 algorithm. According to the results of the blockchain network, it can be seen that a single block size is close to each other under the CR and Timmins datasets. Furthermore, tests under the Timmins dataset leads to a longer time of single block time as 8.59s than CR as 6.56s.

Table 4.3: Blockchain network performance. TSize: total blockchain size (byte), SB: single block, TTime: total time (s), HR:HashRate ((MH/s)

	TSize	SB Size	TTime	SB Time (s)	HR	Number of Blocks
CR	856330	1178	2351	6.56	1.55	201
Timmins	41846	1196	138	8.59	1.65	275

Meanwhile, a higher hash rate is obtained under the Timmins dataset with 1.65 MH/s than 1.55 MH/s in CR dataset. Test results are presented in Table 4.3. However, reaching consensus in a blockchain network requires computation that leads to high cost. Mitigating blockchain costs is an important consideration, especially in public blockchain networks where transaction fees and operational expenses can be significant. Exploring cost-cutting measures for the proposed PBFT-based blockchain system has not been examined yet, but it unquestionably represents a significant area for future research.

Probability of saving a task

The probability of storing a new transaction is presented in (4.14), which depends on the system architecture. From (4.14), we can see the probability of saving a task depends on the scale of a system (represented by the number of nodes N), individual node fault probability (p), and the number of faulty nodes k. We consider N as 5, 10, 15, or 20 and p as 0.1, 0.2, 0.3 to evaluate performance later.

As introduced in (4.11), consensus probability of a PBFT-based system depends on the maximum number of faulty nodes k_{max} under particular system scale N and node fault probability p. The maximum number of faulty nodes k_{max} tolerated by PBFT can be obtained on the basis of (4.2),that is 1, 3, 4, 6 for the blockchain with a total of 5, 10, 15, 20 nodes, respectively. A blockchain network can be scaled by enlarging the number of nodes in the chain. In this thesis, the largest number of nodes in the blockchain network is 20. Meanwhile, according to (4.4),), the number of active nodes is measured as 4, 7, 11, and 14 for the blockchain with a total of 5, 10, 15, 20 nodes, respectively. With a particular node fault probability p, k_{max} and a total number of nodes N, it is straightforward to compute the maximum probability of storing a task in a PBFT-based system using (4.11). Meanwhile, probability of a task being stored in a centralized system and a NFT system is calculated according to (4.14) as shown in Table 4.4. We can see that the higher node fault probability (p) leads to a lower probability of saving a sensing task for all three systems. Meanwhile, a PBFT-based system results in a higher probability of saving than a NFT system. This shows the advantage of PBFT over NFT systems. Especially for large

Table 4.4: Probability of a task being stored in PBFT-based blockchain, centralized system and NFT system. N stands for the number of nodes in a system.

	PBFT (N=5)	NFT (N=5)	PBFT (N=10)	NFT (N=10)	PBFT (N=15)	NFT (N=15)	PBFT (N=20)	NFT (N=20)	Cen (N=1)
p=0.1	0.919	0.590	0.987	0.349	0.987	0.206	0.998	0.122	0.9
p=0.2	0.737	0.328	0.879	0.107	0.836	0.035	0.913	0.012	0.8
p=0.3	0.528	0.168	0.650	0.028	0.515	0.005	0.608	0.001	0.7

scale systems (e.g., N=20), PBFT is significantly superior to its side-effects. For instance, PBFT contributes 0.998 probability while NFT results in 0.122 one when N equals 20 and p is 0.1. When PBFT is compared to a centralized system, PBFT contributes to a higher probability than a centralized one when p is 0.1. When p equals 0.2, a large scale blockchain network (N greater than than 5) outperforms a centralized system. However, a centralized system performs better when node fault p is 0.3. Specifically, a centralized system contributes 0.7 probability when compared to 0.528, 0.65, 0.515, and 0.608 via PBFT-based blockchain under p as 0.3.

Task detection performance

Ensemble learning model is designed to detect fake tasks as introduced in Fig. 4.6. Adaboost, KNN, DT, and SVM algorithms are arranged in four different blocks in the blockchain, which outputs prediction for task legitimacy independently as shown in Fig. 4.6. As introduced in Section 4.2.2, each block contains a copy of the same training set to train the deployed ML model. In this work, these four models assemble the same samples which are collected in PBFT $Request$ and $Pre-Prepare$ messages. The training dataset distribution is described in Table 4.2. Individual ML model prediction is transferred to a server (an Aggregator) to make a decision via simple majority voting. Specifically, if two or more than two ML algorithms predict a sample as legitimate, the Aggregator decides that the sample is legitimate; otherwise, the Aggregator marks the task as fake. Fig. 4.10 demonstrates accuracy by individual ML algorithms and vote-based ensemble learning approach and Table 4.5 presents confusion matrix. The ensemble learning method outperforms the individual ML algorithms by achieving up to 0.99 accuracy whereas Adaboost, KNN, DT, and SVM achieve 0.98, 0.93, 0.96 0.911, respectively, under Timmins dataset. On the other hand, under the CR dataset, the ensemble learning approach demonstrates a detection accuracy of 0.91 more than KNN 0.87 and DT 0.88. However, the ensemble learning method performs worse than SVM 0.96 because SVM predicts all tasks as legitimate

as shown in Table 4.5. The ensemble learning approach in the proposed blockchain-based framework makes decisions based on the results of four legacy ML algorithms. Therefore, the runtime of the ensemble learning comprises individual ML model prediction times and the aggregator's decision-making time. Fig. 4.11 presents the runtime of the proposed method and base ML algorithms. By utilizing powerful computing resources, aggregator decision time is significantly shorter than that of the base ML classifier training and inference time. Therefore, ensemble learning time is approaches the maximum time of the base algorithm, which is 2.6s under the CR dataset and 1.15s under the Timmins dataset. The assumption here for the proposed ensemble learning is that the ensemble learner predicts fake tasks immediately and does not take into account blockchain computation time. It woul be invaluable to consider the latency for blockchain in calculating in the future work.

Regarding energy consumption, a computing node consumes energy as described by the equation

$$W = P \times \Delta_t$$

, where P stands for power and Δ_t denotes the operational duration of the device. For the sake of simplicity, we use the rated power to represent P that is a fixed value for a device. It means traditional ML models and the proposed ensemble method share the same power in measuring energy consumption. Furthermore, under the same P, the operational duration of a device Δ_t can be utilized to compare the energy consumption of various methods based on the energy (W) formulation.

The inference time of the base ML algorithms is presented in Fig. 4.11. However, the proposed ensemble learning depends on four ML models estimation results; hence, it is contributed by all base ML algorithms. The ensemble learning energy consumption is the total of the four ML algorithms. Thus, the time of ensemble learning model is sum of the working durations of four base ML models.

With generic power P and methods working duration in Fig. 4.11, energy consumption is presented in Fig. 4.12. As introduced before in this chapter, a real test environment is built for evaluation using MacBook Pro, that introduces the rated power P as 60.9W. Based on the rated power P and fake task detection time, electricity energy consumed are illustrated in Fig. 4.12. Although the ensemble learning in the proposed approach leads to a higher detection performance, it raises time complexity and energy complexity critically. Specifically, the proposed method consumes energy about 2.5 times to 6.67 times of the conventional MLs. In light of this observation, we identify the issue of addressing the trade-off between training accuracy and energy/delay as an element of our immediate future agenda.

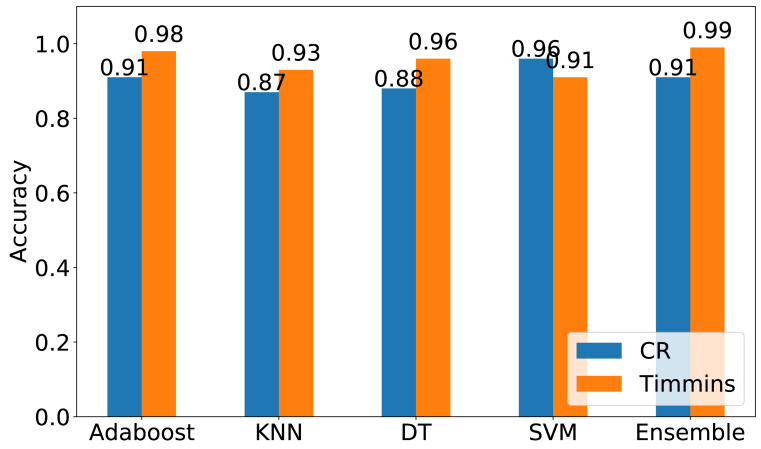


Figure 4.10: Accuracy comparison under CR and Timmins datasets

Table 4.5: Confusion matrix for individual and ensemble learning algorithm under Timmins dataset. PF: predicted fake task, PL: predicted legitimate task.

	Adaboost		KNN		DT		SVM		Ensemble	
Timmins	PF	PL	PF	PL	PF	PL	PF	PL	PF	PL
True Fake	12	3	8	7	10	5	0	15	14	1
True Legitimate	0	153	5	148	1	152	0	153	0	153
CR	PF	PL	PF	PL	PF	PL	PF	PL	PF	PL
True Fake	101	0	78	22	90	11	0	101	99	2
True Legitimate	228	2161	291	2098	279	2110	0	2389	218	2171

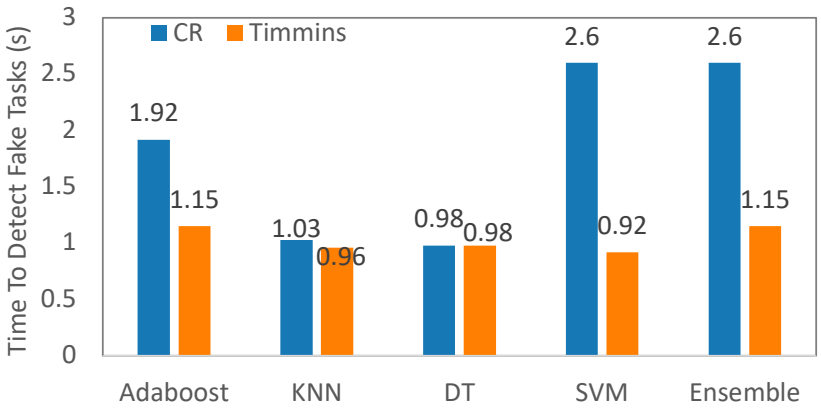


Figure 4.11: Time in detecting fake tasks comparison under CR and Timmins datasets

Performance of tasks saving

RoLT and *RoFT* are criteria to measure legitimate task and fake task saving performance, that are formulated in (4.15) and (4.16), respectively. The MCS platform aims to save more legitimate tasks and less fake tasks. Therefore, it can be said that the proposed system performs well when it performs with high *RoLT* and low *RoFT*. *RoLT* definition in (4.15) implies that it depends on the total number of legitimate tasks T_l, number of correctly predicted legitimate tasks w, and task saving probability p_s. *RoFT* is determined by total number of fake tasks T_f, number of predicted incorrect fake tasks v, and task saving probability p_s. p_s calculation results haven been given in Section 4.2.3 under different system architectures, scales of systems, and the probability of faulty nodes. Three scenarios are considered to compare with the proposed system for *RoLT* and *RoFT*, including **Scenario 1** under different detection approaches, **Scenario 2** under different system architectures (e.g., decentralized or centralized); and **Scenario 3** under fault tolerance systems or non-fault tolerance systems. Table 4.6 demonstrates system configuration in the test for the three scenarios test.

Scenario 1: RoLT and RoFT under different detection approaches A task predicted as legitimate is considered as a new transaction sent to the blockchain, which triggers PBFT consensus procedures as presented in Fig. 4.6. It means different detection approaches with various detection performance impacting tasks saving performance. Task prediction results are demonstrated in the confusion matrix as shown in Table 4.5 and Table 4.5 for Timmins and CR datasets, respectively. For instance, with ensemble learning-based detection, there is 1 fake task (v is 1) and 153 (w is 153) legitimate tasks estimated as

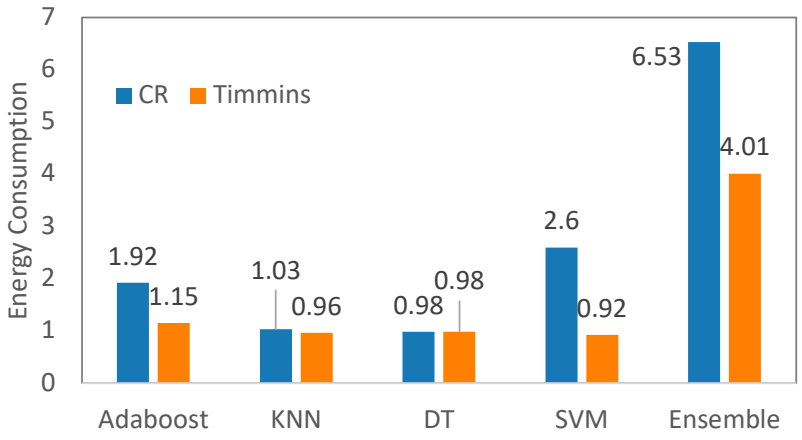
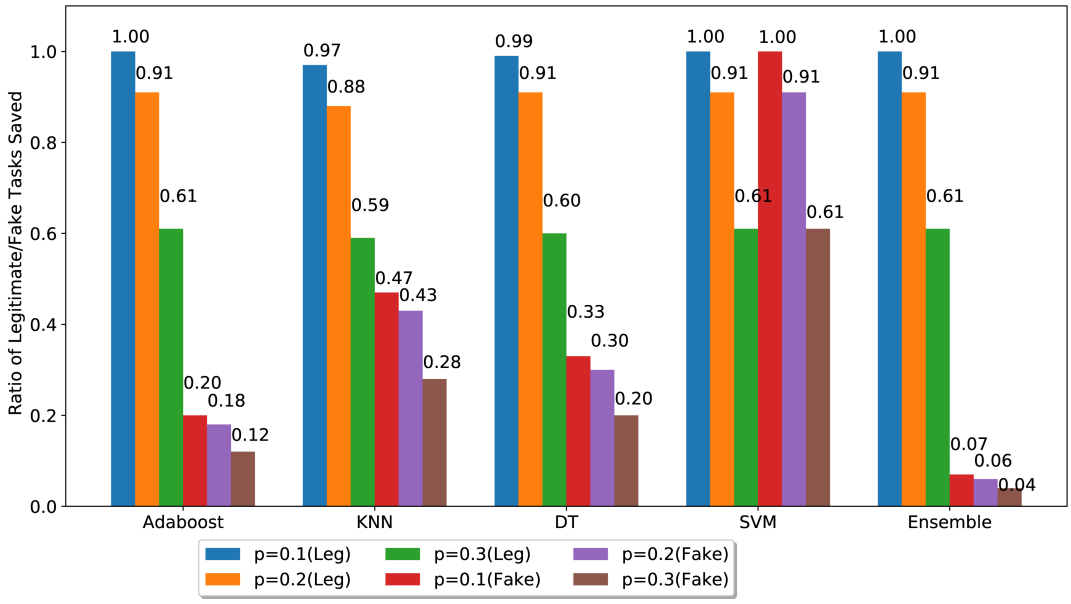


Figure 4.12: Energy consumption comparison under CR and Timmins datasets

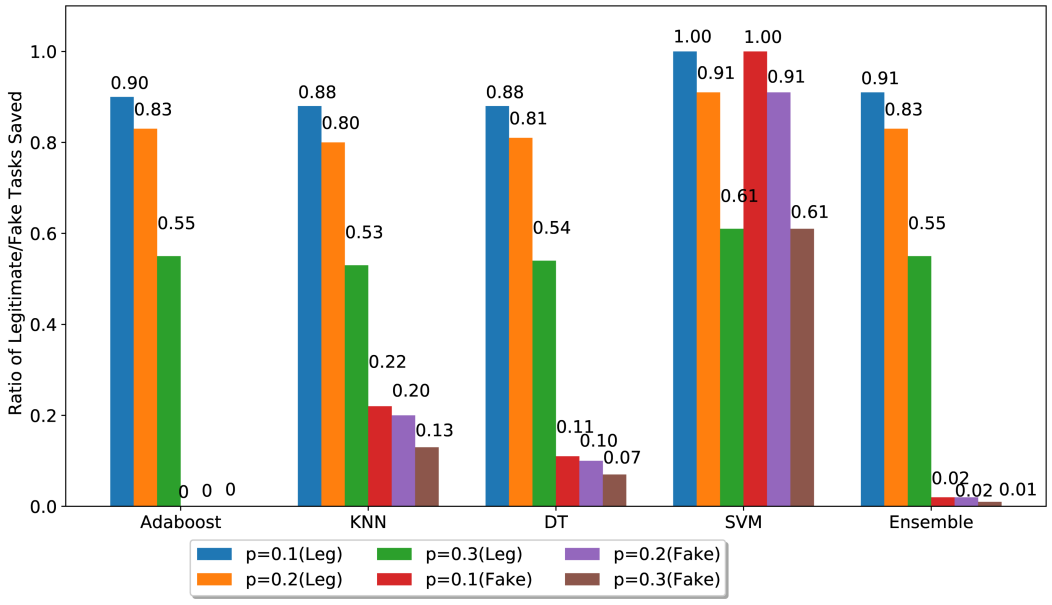
Table 4.6: System configuration under three scenarios for *RoLT* and *RoFT* test. FT: fault tolerance. *Various* denotes system configuration according to test requirements. For example, Various means that various ML algorithms are considered in Scenario 1.

	Scenario 1	**Scenario 2**	**Scenario 3**
Architecture	Decentralized	**Various**	Decentralized
ML	**Various**	Ensemble	Ensemble
FT	PBFT	PBFT	**Various**

legitimate tasks under Timmins dataset and while there are 2 fake tasks (v is 2) and 2171 legitimate tasks (w is 2171) predicted as legitimate ones. Total 154 tasks in Timmins dataset and 2173 tasks in CR dataset will be sent to blockchain-based MCS platform to store. Individual ML algorithms contribute different prediction results as shown in Table 4.5 and Table 4.5. In this test scenario, system architecture is fixed with decentralized blockchain network and PBFT technique is applied as shown in Table 4.6. Here, *RoLT* and *RoFT* are verified under blockchain-based system with various detection approaches (e.g., Adaboost, KNN, DT, SVM and ensemble learning). According to the results of system probability of saving a task in Table 4.4, blockchain with 20 nodes contributes highest probability than results with 5, 10, and 15 so we fix blockchain with 20 nodes to conduct the test with different detection approaches. It is straightforward to calculate



(a) Timmins dataset



(b) CR dataset

Figure 4.13: *RoLT* and *RoFT* under different detection methods. $p = 0.1(Leg)$, $p = 0.2(Leg)$ and $p = 0.3(Leg)$ stand for *RoLT* results while $p = 0.1(Fake)$, $p = 0.2(Fake)$ and $p = 0.3(Fake)$ denote for *RoFT* results

RoLT and *RoFT* with different detection results (v and w obtained). Comparison of *RoLT* and *RoFT* is shown in Fig. 4.13. Legends in Fig. 4.13 denotes individual device fault probability and task legitimacy. For instance, $p = 0.1(Leg)$ represents a single node fault probability 0.1 to save legitimate tasks.

The ensemble learning method reduces *RoFT* saving critically when compared with traditional ML algorithms. For instance, in $p = 0.1$ (fake) scenario, the ensemble learning approach achieves 0.07 *RoFT*, while SVM achieves 1 *RoFT*. It means that ensemble learning prevents most fake tasks stored in blockchain with 0.07 ratio fake tasks saved in the blockchain. Meanwhile, the ensemble learning approach contributes almost all legitimate tasks saving successfully in the blockchain. Furthermore, we can see smaller fault probability results in better performance with higher *RoLT* and lower *RoFT*. For example, ensemble learning model performs 1 *RoLT* under $p = 0.1$, 0.91 under $p = 0.2$, and 0.61 under $p = 0.3$. On the other hand, the results under CR dataset shows a similar trend as the results under Timmins dataset except SVM results. We can see SVM contributes the higher *RoLT* than the ensemble learning approach. However, SVM results in the worst performance *RoFT* among all methods with up to 1 *RoFT* under $p = 0.1$. When compared the results in CR and Timmins datasets, a better *RoLT* and worse *RoFT* are obtained under Timmins dataset than the results using CR dataset. For instance, with $p = 0.1$, the ensemble learning approach achieves *RoLT* of 1 and *RoFT* of 0.07 under Timmins dataset while it performs 0.91 *RoLT* and 0.02 *RoFT* with $p = 0.1$ under CR dataset.

Scenario 2: RoLT under centralized and decentralized systems As introduced in Section 4.2.3, system architecture (centralized or blockchain-driven decentralized system) determines a task save probability so it impacts *RoLT* and *RoFT* directly. As ensemble learning approach outperforms individual ML methods for task detection performance which has been demonstrated in Section 4.2.3. In this test scenario, the ensemble learning approach is applied to filter fake tasks and fault tolerance is considered as shown in Table 4.6. Centralized and decentralized system performance is compared under this test scenario. As *RoFT* by ensemble learning model is extremely low, discussed in Fig. 4.13, the *RoFT* results are not presented in **Scenario 2**. *RoLT* is measured, taking ensemble learning-based prediction results and under a centralized system and blockchain-based decentralized systems. As introduced in Section 4.2.3, ensemble learning model predicts 153 (w) legitimate tasks and 1 (v) fake task as legitimate under the Timmins dataset, and the results are shown in Table 4.5. Meanwhile, ensemble learning model estimates 2171 (w) legitimate tasks and 2 (v) fake tasks as legitimate ones using CR dataset as demonstrated in Table 4.5. In Section 4.2.3, probabilities of PBFT-based blockchain networks and centralized system have been calculated as shown in Table 4.4, which demonstrates results considering different node faulty probability and different scales of a system. Additionally,

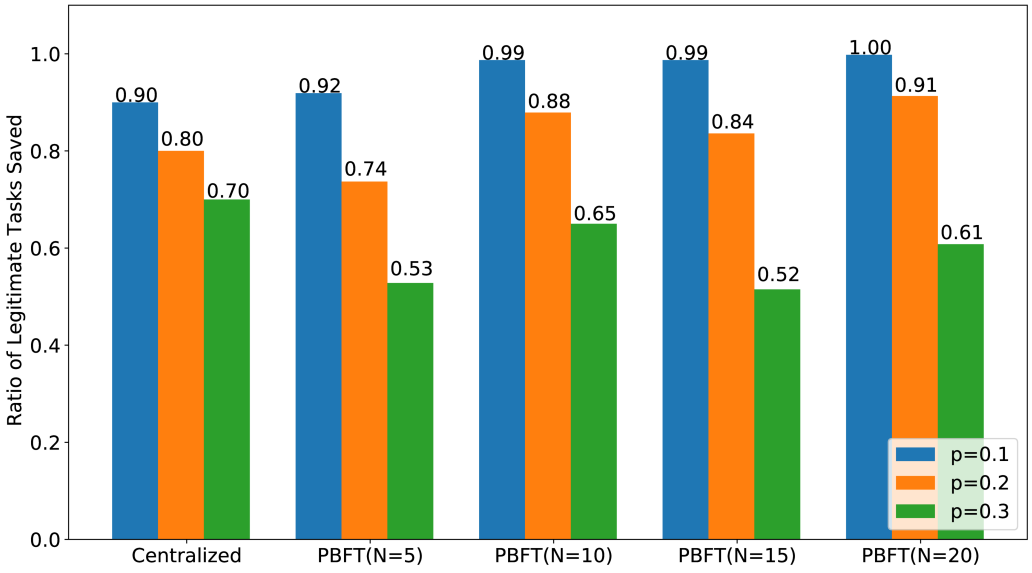
taking ensemble learning model estimation results, $RoLT$ is calculated according to (4.15). The results are illustrated in Fig. 4.14.

The proposed PBFT-based blockchain demonstrates a more promising performance than a centralized system for most test cases. As the number of nodes in a blockchain increases, the proposed PBFT-based systems boosts $RoLT$ for fault probabilities of 0.1 and 0.2 when compared with centralized system. Specifically, with 0.1 fault probability, $RoLT$ increases from 0.835 under 5 nodes to 0.907 under 20 nodes using CR dataset. Regarding to the proposed PBFT-based system, the higher fault probability results in lower $RoLT$. For example, under Timmins dataset, $RoLT$ of PBFT in the 5-node blockchain takes values of 0.919, 0.879 and 0.528 for p values of 0.1, 0.2, and 0.3, respectively. This is reasonable as probability of system failure increase when each node has a higher failure rate. However, the introduced blockchain system performs severe fluctuation under $p = 0.3$, which shows worse performance than a centralized system.

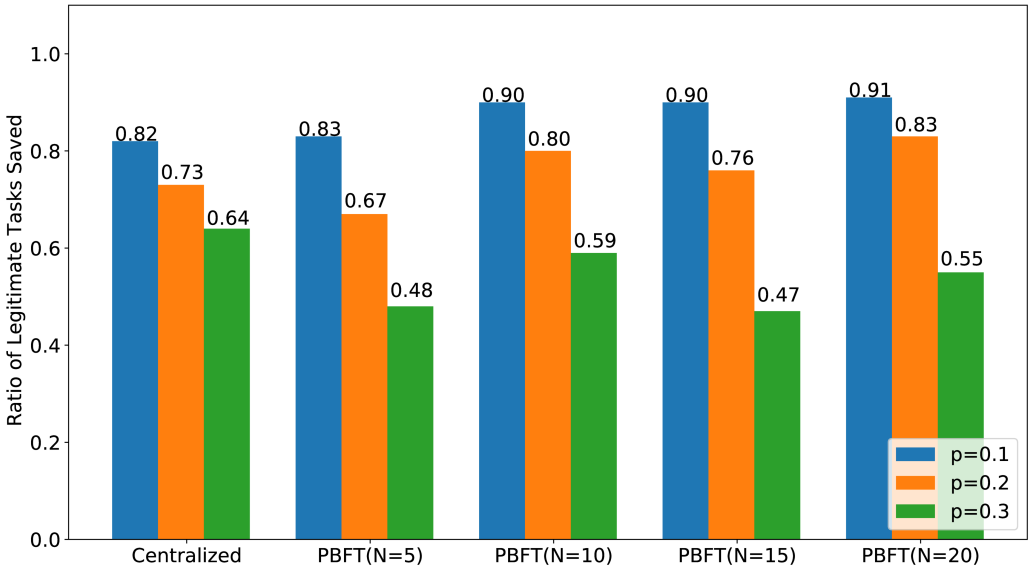
Scenario 3: RoLT and RoFT under fault tolerance and Non-Fault Tolerance (NFT) systems This section compares $RoLT$ under PBFT-based fault tolerance system and NFT system performance, taking ensemble learning-based detection performance the same as **Scenario 2**. As illustrated in (4.12), a NFT system requires all nodes active and working well. A NFT system with various nodes working probability is presented in Table 4.4 for N as 5, 10, 15, 20. Similarly, $RoFT$ is ignored here due to ensemble learning-based prediction results applied in PBFT-based system and NFT systems. Based on the ensemble learning model prediction results and system saving a tasks probability in Table 4.4 in Section 4.2.3, $RoLT$ is calculated for PBFT-based fault tolerance system and the NFT system as described in Fig. 4.15. It indicates that PBFT-based system boosts dramatically when compared with the NFT system for test cases. Mainly, when node fault probability equals 0.3, NFT performs severely, with almost 0 $RoLT$ in the NFT system while PBFT contributes 0.61 $RoLT$ in Timmins and 0.55 $RoLT$ in CR. In PBFT-based system, increasing the number of nodes, $RoLT$ raises gradually. However, $RoLT$ decreases critically when raising the number of nodes. Especially, for 20 nodes system, NFT system performs 0 to 0.12 $RoLT$ while PBFT-based system obtains 0.61 to 1.00 $RoLT$ under Timmins dataset. Meanwhile, results in the two dataset almost share the same tendency.

4.2.4 Conclusion

A MCS system is conventionally implemented as a centralized one which faces single point of failure. Nevertheless, this centralized MCS system has important issues such as sensing task tampering. This topic is rarely investigated in current work to protect sensing task

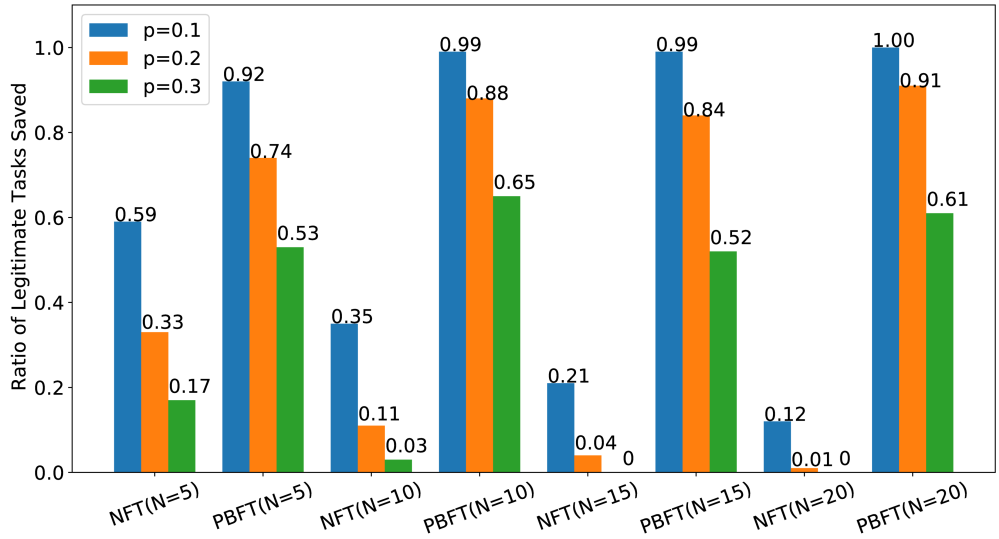


(a) Timmins dataset

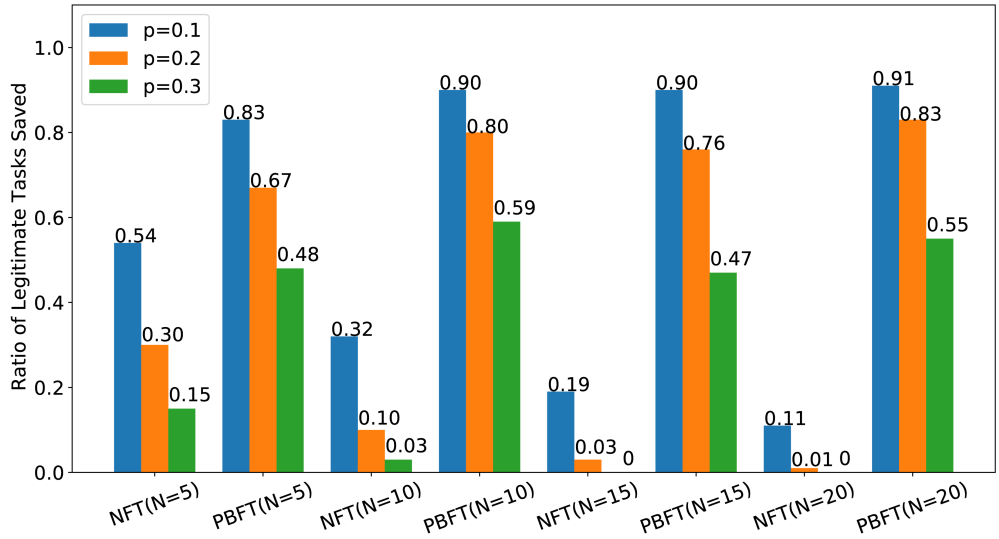


(b) CR dataset

Figure 4.14: *RoLT* under centralized system and PBFT-based decentralized system



(a) Timmins dataset



(b) CR dataset

Figure 4.15: *RoLT* comparison of Non-Fault Tolerance (NFT) and PBFT protocol

avoiding revisions. Meanwhile, the fake sensing task attack impacts the MCS platform and participants by draining more energy from smart equipment and clogging sensing sever. this work introduces a distributed decentralized MCS framework integrating blockchain technique to address data tempering and DoS threads. The blockchain network consists of PBFT consensus to determine a transaction can be added to not in a blockchain. As PBFT tolerates several Byzantine faulty nodes, the designed MCS system becomes robust to resist faults. The decentralized system avoids task requesters submitting tasks to a centralized MCS platform, which boosts security level of the system and protects requester's data to some extend. Meanwhile, the ensemble learning approach is designed in the blockchain-based MCS platform to detect and eliminate fake tasks that prevents fake task introduced DoS attack. Furthermore, our proposed approach endures individual node fault probability to achieve a high Rate of Legitimate tasks ($RoLT$) and a low Rate of Fake tasks ($RoFT$) when compared with a centralized system and Non-Fault Tolerance (NFT) system.

In the performance evaluation section, two different datasets are generated via CrowdSenSim Tool under two real maps for a big city Clearance-Rockland (CR) and a small town (Timmins). Regarding to task detection performance, the ensemble learning algorithm prevents most fake tasks with up to 0.99 and 0.91 accuracy under Timmins and CR dataset, respectively, that outperform classic machine learning (ML) algorithms. Since we need to save sensing tasks into blockchain network to avoid tampering, tasks saving performance is critical to verify. In this work, Ratio of Legitimate Tasks ($RoLT$) and Ratio of Fake Tasks ($RoFT$) are considered. The ensemble learning-integrated system contributes higher $RoLT$ and lower $RoFT$ than all individual ML algorithm-based system. For instance, ensemble learning-based system obtains 0.07 $RoFT$ while Adaboost-based system performs 0.20 $RoFT$ using Timmins dataset with node fault probability as 0.1. Moreover, the proposed decentralized system outperforms a conventional centralized system, in terms of $RoLT$, especially for small node fault probability. Specifically, the introduced system obtain 0.91 $RoLT$ while the centralized system performs 0.82 $RoLT$ under CR dataset with 0.1 node fault probability. Furthermore, the proposed PBFT-based decentralized system contributes a more promising performance ($RoLT$) than NFT-based decentralized system. For example, numerical results demonstrate 0.91 $RoLT$ in PBFT-based system and 0.11 in NFT-based system under 20 nodes in total and node fault probability 0.1.

Chapter 5

Network Intrusion Detection Via Machine Learning Algorithms in IoT

IoT has been an emerging paradigm to interconnect a plethora of sensors and uniquely addressable devices to exchange data directly without human intervention [117, 19]. The essential purpose of IoT is to contribute a network framework with general protocols and software to enable connecting and incorporating of all connected nodes in the network [8, 55]. For instance, an IoT-based smart home consists of many devices (e.g., smart appliances, smart locks, smart hubs) to collect and monitor various data types in our daily life [221]. Due to their large-scale deployment and ubiquity, IoT systems confront critical cybersecurity threats, among which network intrusion appears to be one of the top vulnerabilities as a massive number of connected devices lead to a giant attack surface in IoT systems [116].

In order to address network intrusion, NIDS is typically deployed in IoT systems. A NIDS is used to detect potential perspective attacks (e.g., malicious operations, various attacks) and notify the appropriate persons based on detection results. HIDS is meritorious as it detects anomalies without analyzing or monitoring the network traffic. The authors in [202] introduce an HIDS use case to secure Android mobile equipment. HIDS is beneficial as it offers fine-grained solution to detect anomalous patterns internally [150]. Therefore, the role of HIDS in the detection of ATP is crucial. The authors in [165] show that APT is detectable via HIDS in an efficient manner.

In this chapter, several public datasets are introduced. Moreover, two ML-based NIDS frameworks, namely Prediction of the Lowest Cost (POLC) and All Predict Wisest Decides (APWD), are introduced, that make wise prediction based on prediction performance and

costs, respectively; and a hybrid framework integrating NIDS and HIDS is demonstrated that boost intrusion prediction accuracy further.

Table 5.1: NSL-KDD dataset distribution for training and testing [223]

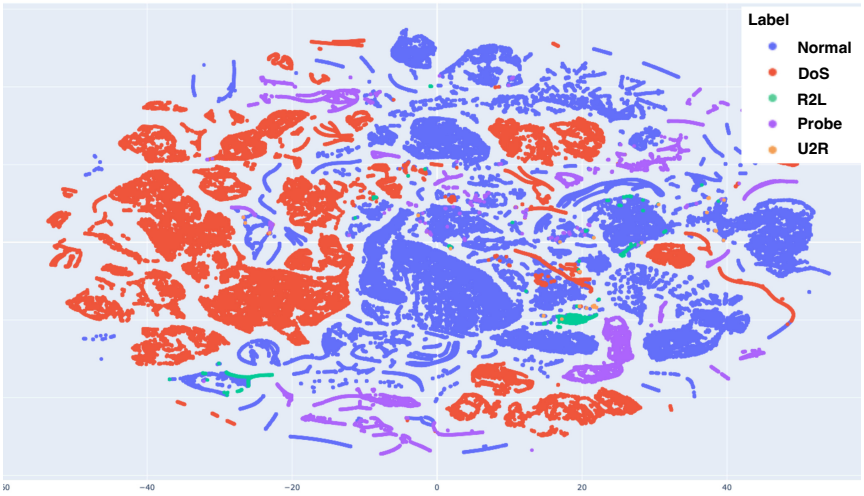
Trainig dataset		Testing dataset	
Class	Patterns	Class	Patterns
Normal	67343	Normal	9711
DoS	45927	DoS	7458
Probe	11656	Probe	2421
R2L	995	R2L	2754
U2R	52	U2R	200
Total	125973	**Total**	22544

5.1 Dataset Introduction

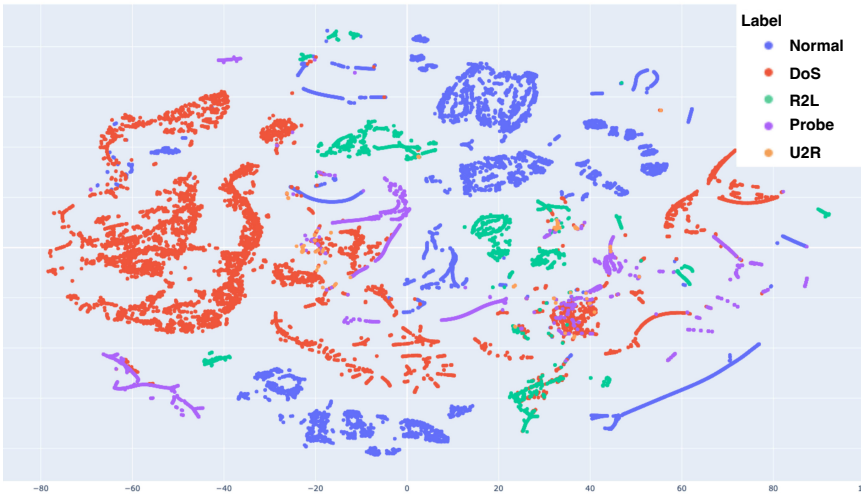
Four public datasets are utilized in evaluating the presented methods performance, including NSL-KDD, SCVIC-APT-2021, CICIDS 2018, and NDSec-1.

5.1.1 NSL-KDD

NSL-KDD [60] is a popular dataset that is one of the most used datasets in the intrusion detection system. It consists of four types of attacks and normal traffic data. This dataset is widely applied as a practical benchmark dataset to help researchers compare different intrusion detection methods [263]. Table 5.1 presents the NSL-KDD dataset classes and the number of instances for each category. From the NSL-KDD class distribution, it is an extremely balanced dataset. Normal traffic and DoS attack data dominate in the training dataset and testing dataset. On the other hand, R2L and U2R have 0.046% and 0.88% proportion rates in the NSL-KDD training dataset. The NSL-KDD dataset visualization is demonstrated in Fig.5.1 using the t-SNE method [154]. It shows test dataset contains several new attacks that only exist in the test dataset; i.e., a ML model cannot learn from the training dataset. It is challenging for MLs to obtain high detection performance via training itself using training dataset.



(a) Training dataset



(b) Testing dataset

Figure 5.1: NSL-KDD dataset visualization using t-SNE

Table 5.2: Training and test partitions of the SCVIC-APT-2021 dataset

Trainig dataset		Test set	
Class	Patterns	Class	Patterns
DataExfiltration	601	DataExfiltration	74
InitialAttack	150	InitialAttack	77
LateralMovement	871	LateralMovement	142
NormalTraffic	208686	NormalTraffic	36476
Pivoting	2482	Pivoting	360
Reconnaissance	1084	Reconnaissance	251
Total	213874	**Total**	37380

5.1.2 SCVIC-APT-2021

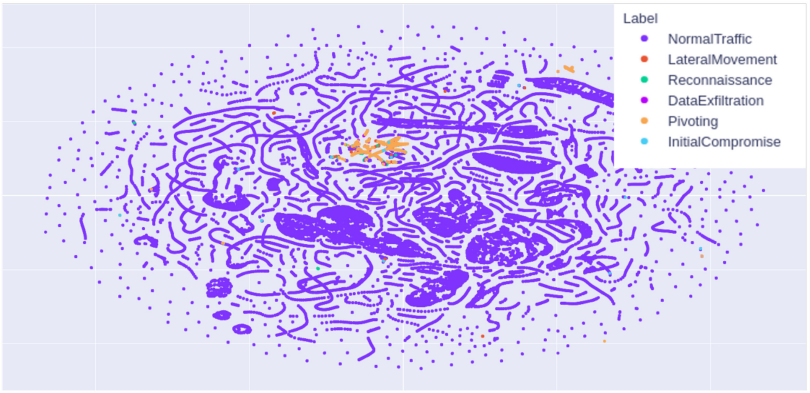
The SCVIC-APT-2021 dataset includes six attack types (e.g., Pivoting, Lateral Movement, Data Exfiltration, Reconnaissance and DoS) and normal traffic, which bridges the gap between rare public datasets consisting of these attacks. [1].

Table 5.2 illustrates the distribution of training and test datasets. Both of them are significantly imbalanced and overwhelmingly dominated by normal traffic data. Thus, the distribution is similar to a real network intrusion dataset. Fig.5.2 visualizes the dataset (i.e., training and test partitions). This figure shows that normal traffic and attack samples have different distributions. However, attack traffic data is mixed and overlapped as shown in Fig.5.2. Thus, classification of multiple attacks could be challenging.

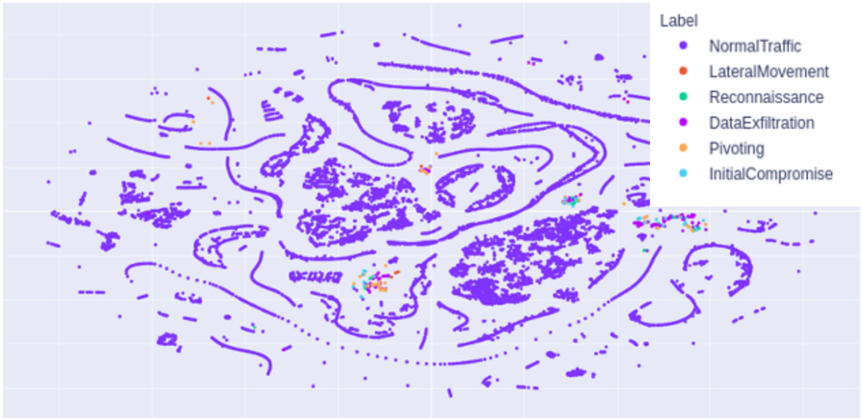
5.1.3 CICIDS 2018

CICIDS 2018 dataset is a public dataset [2] that dataset contains seven types of attacks, including Brute-force, Heartbleed, Botnet, DoS, DDoS, Web attacks, and infiltration of the network from inside. The attacking infrastructure contains 50 equipment, and the victim system has 5 branches and consists of 420 devices and 30 servers. The CICIDS 2018 dataset comprises the captured network traffic and recorded system logs of each device, along with 80 features obtained from the captured traffic [107]. The seven attack scenarios are categorized into 14 types, including Bot, DDOS-HOIC, DDOS-LOIC-HTTP, DDOS-LOIC-UDP, DoS-GoldenEye, DoS-Hulk, DoS-SlowHTTPTest, DoS-Slowingloris,

[1]SCVIC-APT-2021 dataset can be downloaded via this link https://dx.doi.org/10.21227/g2z5-ep97
[2]The dataset is introduced in detail at: https://www.unb.ca/cic/datasets/ids-2018.html



(a) Training dataset



(b) Testing dataset

Figure 5.2: SCVIC-APT-2021 dataset visualization using t-SNE

Table 5.3: CICIDS 2018 dataset sample distribution

Class	Training	Test	**Total**
Benign	308375	152172	**460547**
Bot	60767	29693	**90460**
DDOS-HOIC	137147	67449	**204596**
DDOS-LOIC-HTTP	39019	19166	**58185**
DDOS-LOIC-UDP	760	342	**1102**
DoS-GoldenEye	2271	1163	**3434**
DoS-Hulk	13388	6553	**19941**
DoS-SlowHTTPTest	10579	5351	**15930**
DoS-Slowloris	1394	702	**2096**
FTP-BruteForce	32222	15918	**48140**
SSH-Bruteforce	11143	5418	**16561**
Brute Force-Web	27	15	**42**
Brute Force-XSS	10	6	**16**
Infiltration	21	8	**29**
SQL Injection	17	8	**15**

FTP-BruteForce, SSH-Bruteforce, Brute Force-Web, Brute Force-XSS, Infiltration, and SQL Injection.

5.1.4 NDSec-1

The NDSec-1 dataset was initially introduced in [26], and was generated for an attack composition in network security research schemes. The NDSec-1 dataset is a public dataset which is used in network intrusion detection [174, 137]. The dataset consists of 8 types of attacks including Botnet, Bruteforce, DoS, Exploit, Malware, Misc, Probe, and Webattack, Spoofing and benign samples. However, there is only one sample for the Spoofing class, so this class is excluded in the evaluation section. Sample distribution across training and test sets is presented in Table 5.4. Flow features are extracted from the NDSec-1 dataset and included with 63 network features. Meanwhile, log records in hosts are provided, so host features are extracted using Bert for text to numeric array transforming. It adopts the CICIDS 2018 dataset host features extraction procedures. Finally, the NDSec-1 dataset contains two parts of host features, including two-dimensional event features and two-dimensional message features as shown in Fig. 5.13. Table 5.23 demonstrates NDSec-1

Table 5.4: NDSec-1 dataset sample distribution

Class	Training	Testing	**Total**
BOTNET	52	40	**92**
BRUTEFORCE	2150	1006	**3156**
DOS	12677	6389	**19066**
EXPLOIT	4	2	**6**
MALWARE	23	12	**35**
MISC	31	18	**49**
NORMAL	5701	2284	**7985**
PROBE	777	307	**1084**
WEBATTACK	18	12	**30**

dataset consisting 63 network features, $2182 * 8$ event features and $512 * 768$ message features.

5.2 Network Intrusion Detection Via Machine Learning Algorithms in IoT

Although ML algorithms are widely deployed against network intrusion, their performance on identifying each intrusion type varies. Thus, under the same test dataset, it is possible to observe an ML model (e.g., model_1) perform better for an attack-type (e.g., class_1), and another model (e.g., model_2) outperform the former for other classes (e.g., class_2 and class_3). Based on this motivation, the APWD method is proposed. APWD is a general ensemble method that can be used as benchmark methodology to integrate various ML models. Under the APWD framework, multiple ML models are independently trained and tested to determine the wisest model for each attack type (i.e., highest F1 score). The aggregation model makes the final decision relying on the model that has been identified as the wisest for a given class. In order to simplify the evaluation process, three ML models (e.g., RF, XGBoost, Adaboost) are chosen as the base ML models for APWD. The numerical results show the APWD framework demonstrates not only the benefits of individual class performance-boosting but also improves overall performance. A public dataset NSL-KDD [193, 223, 145] is used to verify the proposed method APWD, and numerical results show that the APWD framework ensures promising improvements over an individual model and can even achieve up to 18 times the higher performance for R2L

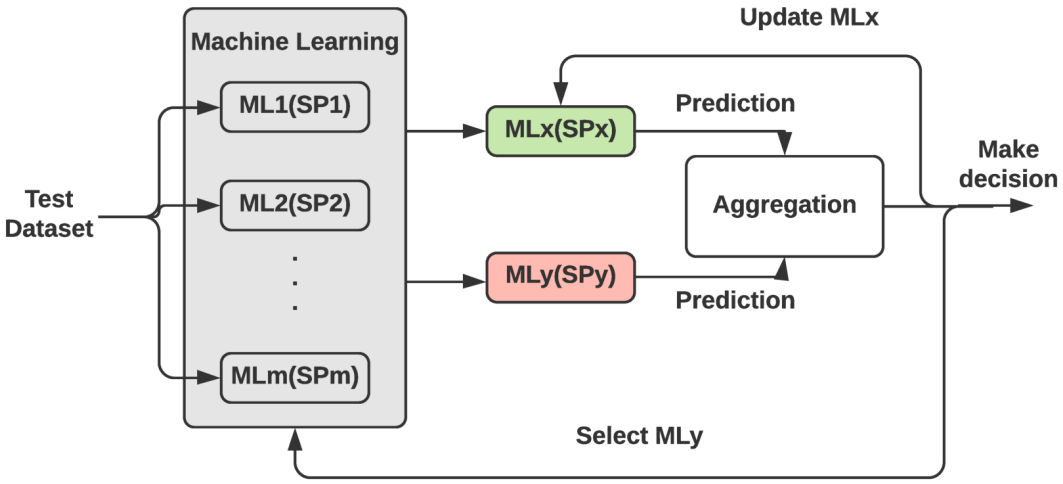


Figure 5.3: APWD system and aggregation model. MLx with the maximum number of highest F1 score and MLy with the second maximum number of highest F1 score

attacks under the NSL-KDD dataset. Meanwhile, the overall accuracy is increased to 0.797 by APWD, in comparison to RF (0.758), XGBoost (0.772), and Adaboost (0.584).

Table 5.5: Matrix of ML models performance for APWD method

		Machine Learning Model F1 Scores For Each Class					
		ML1	**ML2**	**...**	**MLi**	**...**	**MLm**
Pred	**P1**	ML1_P1	ML2_P1	...	MLi_P1	...	MLm_P1
	P2	ML1_P2	ML2_P2	...	MLi_P2	...	MLm_P2
	...	...	...	...	...	...	...
	Pj	ML1_Pj	ML2_Pj	...	MLi_Pj	...	MLm_Pj
	...	...	...	...	...	...	...
	Pn	ML1_Pn	ML2_Pn	...	MLi_Pn	...	MLm_Pn
F1_max_num		**ML1_F1 _max_num**	**ML2_F1 _max_num**	**...**	**MLi_F1 _max_num**	**...**	**MLm_F1 _max_num**
Set of groups with highest F1 score in ML		**SP1 = {ML1_Pj}, if ML1_Pj >MLt_Pj (t = 2,3,..m)**	**SP2 = {ML2_Pj}, if ML2_Pj >MLt_Pj (t =1,3,..m)**	**...**	**SPi = {MLi_Pj}, if MLi_Pj >MLt_Pj (t =1,2,..,m, t ≠ i)**	**...**	**SPm = {MLm_Pj}, if MLm_Pj >MLt_Pj (t =1,3,..m-1)**
ML prediction value for every sample in test dataset		$R^{X_k}_{ML1}$	$R^{X_k}_{ML2}$		$R^{X_k}_{MLi}$		$R^{X_k}_{MLm}$

5.2.1 System overview

APWD system overview is presented in Fig.5.3, which shows m machine learning models (e.g., $ML1$,...MLm) that are selected for training and testing. An ML model that performs the highest F1 score for a certain attack type compared to the other models is stored in set

SP (e.g., the classes that ML1 performs the highest F1 score are stored in SP1). MLx is the ML model that contributes to the SP of the maximum size. MLy is the ML model that leads to the SP of maximum size when excluding MLx and corresponding SP (i.e., SPx) are excluded. The aggregation model trusts MLy model for the classes in SPy and MLx for classes in SPx. In the first iteration, a sample in test dataset is picked, and a flag is set as False initially for this sample. After that, read MLx and MLy prediction results for this sample. If MLy estimates this sample belongs to an expertise set of classes of MLy, trust MLy prediction value and set this sample flag as True. If MLy does not show expertise for this sample, check MLx prediction value. If MLx shows estimation result belongs to its expertise set of classes, trust MLx prediction value and set the flag as True. Then check MLx prediction value for this sample. If MLx equals Normal, trust MLx prediction but do not set the flag to True. The flag is used to record which sample has been operated in this iteration when determining expertise. As Fig. 5.4 illustrates, more SPys are selected until all classes have been covered. It means a second or third iteration may need to combine the second or third SPy into the aggregation process. The second iteration process uses an updated MLx in the first iteration and loops all samples in the test dataset. When the new MLy shows expertise for this sample and no flag is added, MLy prediction results are trusted. Otherwise, the MLx decision in the first iteration is retained.

Table 5.6: Notation

Notations	Description
MLi_Pj	F1 score of ML model i for class j
$MLi\_F1\_max\_num$	Number of classes where F1 score under ML model i ranks first
MLx	ML model with the maximum $MLi\_F1\_max\_num$ value
MLy	ML model with the maximum $MLi\_F1\_max\_num$ value after MLx
SPx	Set of classes where F1 score under MLx ranks the first
SPy	Set of classes where F1 score under MLy ranks the first
SPa	$SPa = MLx \cup MLy$, union of set MLx and MLy
n	Total number of classes in a dataset
N	Total number of instances in a test dataset
m	Total number of ML models
X_k	The kth instance in a test dataset
$R^{X_k}_{MLx}$	Prediction value of MLx for the kth instance
$R^{X_k}_{MLy}$	Prediction value of MLy for the kth instance
R^{X_k}	Aggregation model decision for the kth instance=

With notations description in Table 5.6, Fig.5.4 shows a detailed procedure to determine

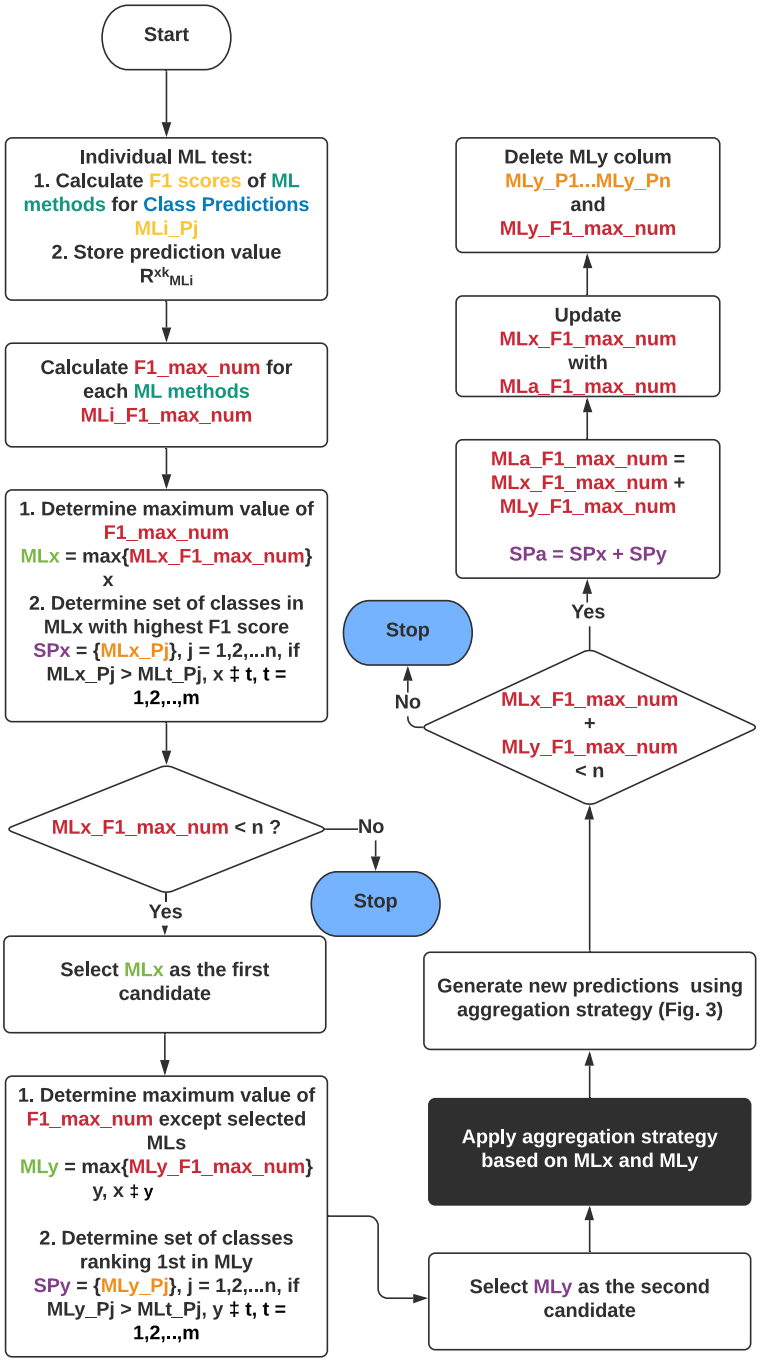


Figure 5.4: Flow of the proposed APWD procedure

the candidate models (i.e., MLx and MLy). It describes candidate selection among m ML models for a dataset with n classes. The APWD workflow is formed by the following eight key steps and related information is summarized in Table 5.5.

- *Step 1: Obtain ML models performance.* Calculate F1 scores of ML methods for each class (MLi_Pj to represent F1 score for ML model i for class Pj), and store prediction values by ML methods for each instance ($R_{MLi}^{x_k}$ to denote the prediction of ML i for the kth instance).

- *Step 2: Sort F1 scores.* Sort F1 scores of ML models for each class and store the number of F1 scores that rank 1st for each ML model;

- *Step 3: Analysis F1 scores sorting results.* Determine the ML model (e.g., MLx) with the maximum number of F1 scores for classes ranking first; and the set SPx is to store the classes where F1 scores under MLx rank the first;

- *Step 4: Check if MLx dominates F1 scores in all classes.* If MLx obtains the highest F1 scores compared to the other ML models, the selected ML models are not suitable for APWD architecture for a further test; Otherwise, choose the model MLx as the first candidate;

- *Step 5: Find the second candidate.* Determine the ML model (e.g., MLy) with the second maximum number of classes ranking first; and the set SPy is to store the classes where F1 scores under MLy rank the first;

- *Step 6: Aggregation based on the two candidate MLs.* Aggregation model makes the final decision for the classes in SPx and SPy;

- *Step 7: Check if further ML models are needed.* If the total number of the classes in $SPx \cup SPy$ is less than the total number of classes in the dataset, select one more ML model excluding MLx and MLy; If additional MLy is needed, a new aggregation process is required. Otherwise, end the procedure;

- *Step 8: Select one more candidate if needed.* Select one more candidate excluding MLx and MLy; update MLx with a combination of MLx and MLy; repeat Step 5 to Step 7 until the aggregation model is able to predict all classes in the dataset;

From Fig. 5.4, MLx achieves the highest F1 scores for the classes in set SPx, and the classes where MLy outperforms the rest are stored in SPy. The aggregation rule is to trust the wisest model decision. Thus, MLy prediction results of classes in SPy are trusted.

Otherwise, the decision is the same as predicting the MLx model for each instance in the test dataset. Based on the rule, the aggregation model makes a decision for each instance in the test dataset.

5.2.2 Performance evaluation

In this section, experimental test cases are conducted for the APWD system, and numerical results under the proposed framework are compared to an individual ML model (e.g., RF, XGBoost, and Adaboost). NSL-KDD dataset introduced in Section 5.1 is used to evaluate the APWD system performance.

ML models including RF, XGBoost, and Adaboost are applied for performance evaluation under the NSL-KDD dataset. Based on the individual model test performance, APWD is applied for further experiments. Tables 5.10 demonstrate RF, XGBoost and Adaboost results. XGBoost outperforms Adaboost and RF in terms of the overall accuracy of 0.772. Based on the individual ML model prediction results, the highest F1 scores for each class are marked with $*$ in Table 5.10, which is candidate selection results of APWD. RF achieves the highest F1 scores for Probe, whereas Adaboost achieves the highest-ranked F1 scores for R2L and U2R. Meanwhile, the best F1 scores for DoS and Normal are achieved by XGBoost. According to above analysis, Table 5.5 is represented as Table 5.7 by filling RF, XGBoost, and Adaboost test results. According to the working process in Fig. 5.4, two iterations are needed in the aggregation procedure.

Table 5.7: Matrix of RF, XGBoost and Adaboost models performance for NSL-KDD dataset

		Machine Learning Model F1 Score For Each Class		
		RF	XGBoost	Adaboost
Class prediction	DoS	0.877	0.889	0.400
	Normal	0.783	0.797	0.738
	Probe	0.749	0.746	0.383
	R2L	0.022	0.101	0.416
	U2R	0.020	0.057	0.183
F1_max_num		1	2	2
Set of groups with highest F1 score in ML		SP1 = {Probe}	SP2 = {DoS,Normal}	SP3 = {R2L,U2R}

In the first iteration, Adaboost and XGBoost are selected for aggregation, and the decision is made for classes DoS, Normal, R2L, and U2R in this step. Furthermore, in the second iteration step, RF is selected for aggregation with motivation to ensure APWD aggregation covering Probe class. Table 5.7 demonstrates parameters configuration results, showing in the main flow in Fig. 5.4, according to selection results.

Table 5.8 illustrates an example of decision making for five samples in the first iteration step for aggregation. APWD relies on XGBoost as MLx for DoS and Normal class prediction and Adaboost as MLy for U2R and R2L prediction. From column Aggr_1, it can be seen that XGBoost-based prediction outputs DoS, DoS, and Normal for samples 0, 1, and 2, respectively. XGBoost exhibits expertise in Normal and DoS patterns, so APWD trusts XGBoost prediction in the aggregation. Therefore, in Aggr_1 column, the sample 0, 1, and 2 decisions are the same as XGBoost estimation. Sample 3 and 4 aggregation results are the same as Adaboost prediction since Adaboost-based prediction results in U2R and R2L for samples 3 and 4, respectively. As introduced before, Adaboost demonstrates expertise in predicting the U2R and R2L traffic patterns. There is a conflict for samples 3 and 4 because XGBoost and Adaboost demonstrate expertise in their prediction. Specifically, XGBoost estimates sample 3 as Normal whereas Adaboost estimates sample 3 as U2R. APWD orders the reputation values of MLx and MLy according to the overall accuracy. APWD trusts the ML model of higher reputation (except for the Normal class) when encountering conflicts. Normal traffic is the majority in the test dataset (and in real time, as well) so ML models would contribute a number of false predictions. Therefore, an ML model (e.g., XGBoost) resulting in higher overall accuracy in Normal class has lower priority than the ML model (e.g., Adaboost) with lower overall accuracy. For sample 3 aggregation, although XGBoost performs at a higher overall accuracy when compared to Adaboost, APWD relies on the Adaboost-based prediction due to XGBoost's estimation that results in the Normal pattern. Following upon the first iteration, MLx prediction value is updated as the aggregation result. The new MLx' is used in the second iteration as shown in Table 5.9 where the same rule in the first iteration is followed.

APWD merging three ML models detection performance is demonstrated in Table 5.11. The proposed approach APWD overall accuracy is increased to 0.7969, which is the highest, compared to 0.7579 under RF, 0.7717 under XGBoost, and 0.5844 under Adaboost. Fig. 5.5 illustrates F1 score comparison between the individual model and APWD. Table 5.12 shows the comparison results between APWD and individual ML models. APWD demonstrates F1 scores no less than all classes when compared to RF, Adaboost, and XGBoost. F1 score for R2L is improved by eighteen times under the proposed method, where the F1 score for the U2L is improved by eight times under APWD compared to RF. Meanwhile, APWD boosts DoS and Probe F1 scores by 122% and 98.4%, respectively,

Table 5.8: An example of the first iteration in aggregation procedure

		MLx	MLy	MLx'
Sample	Label	XGBoost prediction	Adaboost prediction	Aggr_1
0	DoS	DoS	Normal	DoS
1	Normal	DoS	Probe	DoS
2	Probe	Normal	Normal	Normal
3	U2R	Normal	U2R	U2R
4	R2L	Normal	R2L	R2L
...	...	...	...	...

Table 5.9: An example of the second iteration in aggregation procedure

		MLx'	MLy'	Final Decision
Sample	Label	Aggr_1	RF prediction	Aggr_2
0	DoS	DoS	Normal	DoS
1	Normal	DoS	Probe	DoS
2	Probe	Normal	Probe	Probe
3	U2R	U2R	Normal	U2R
4	R2L	R2L	Normal	R2L
...	...	...	...	...

compared to Adaboost. APWD outperforms XGBoost in all classes except DoS, especially obtaining critical improvements for R2L and U2R, where F1 score performance is boosted by three and two times, respectively. APWD obtains the same F1 score 0.889 as XGBoost.

5.2.3 Result analysis

Numerical results demonstrate the proposed APWD framework significantly increases the overall accuracy compared to all base detectors. Experiment results demonstrate the proposed approach APWD boosting overall accuracy to 0.797, comparing 0.772 by XGBoost, 0.758 by RF, and 0.584 by Adaboost. Moreover, in certain attack types R2L, APWD increases F1 score by a factor of 18, from 0.022 by RF to 0.421

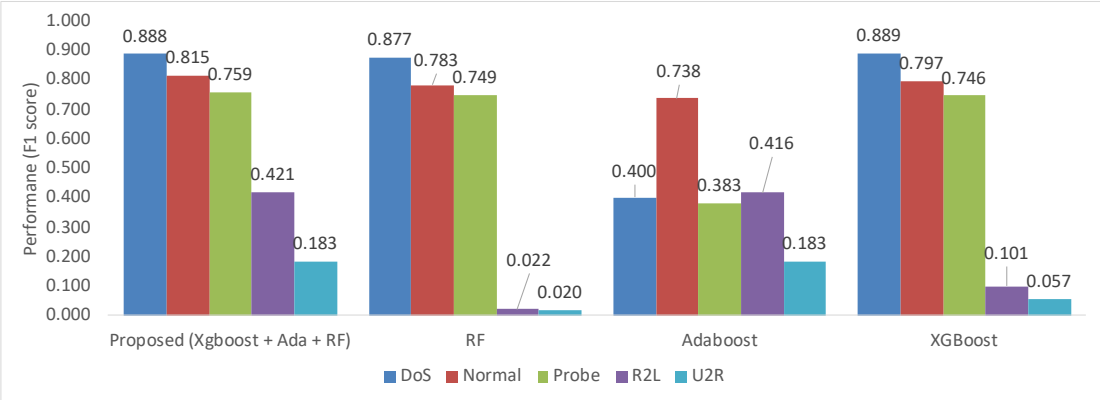


Figure 5.5: F1 score comparison of APWD vs individual models

5.3 New Ensemble Methods for Accurate and Low-Cost Detection of Intrusive Traffic in Networks

This part is an extension of APWD method as introduced in Sec.5.2. APWD consists of multiple classic ML classifiers (namely base estimators, base classifiers, or base detectors) and an aggregation module. With the APWD system, ML algorithms train and test individually to discover the wisest algorithm for every class via checking F1 scores, as shown in Fig. 5.4. Since every attack results in specific damage and a response cost, in this section, an ensemble approach POLC as seen in Fig. 5.7. It extends APWD by using the incurred cost of a decision made by a classifier rather than relying on the F1 score. Both APWD and POLC select expert MLs (MLx and MLy) for aggregation. Cost is considered to select wise ML models to make wise prediction so POLC is proposed in this section.

5.3.1 System overview

As network related dataset is imbalanced that consists of limited number of attack samples such as NSL KDD dataset in Table 5.1. This section introduces GAN-based data augmentation for $R2L$ and $U2R$ attack types to address the insufficient samples. Meanwhile, POLC working procedure are described in detail which both consist of three phases: 1) individual ML model test, 2) expertise determination, and 3) aggregation.

Table 5.10: RF, XGBoost, Adaboot performance comparison, including precision, recall and F1 score, using NSL-KDD. Overall accuracy for RF, XGBoost and Adaboost is 0.758, 0.772, and 0.584, respectively.

Metric	ML	DoS	Normal	Probe	R2L	U2R
Prec	RF	0.963	0.655	0.871	0.938	0.667
	Xgb	0.962	0.675	0.822	0.967	0.600
	Ada	0.845	0.603	0.305	0.912	0.310
Recall	RF	0.805	0.974	0.657	0.011	0.010
	Xgb	0.826	0.971	0.683	0.053	0.030
	Ada	0.262	0.948	0.513	0.270	0.130
F1	RF	0.877	0.783	0.749*	0.022	0.020
	Xgb	0.889*	0.797*	0.746	0.101	0.057
	Ada	0.400	0.738	0.383	0.416*	0.183*

Table 5.11: APWD performance combining XGBoost, RF Adaboost under the NSL-KDD dataset

	Precision	Recall	F1
DoS	0.962	0.826	0.889
Normal	0.709	0.960	0.815
Probe	0.825	0.701	0.760
R2L	0.908	0.274	0.421
U2R	0.310	0.130	0.183
Accuracy	0.797		

Introduction of POLC

The working principle of POLC is quite similar to APWD, which utilizes the average lowest cost instead of the highest F1 score as the indicator for selection. Specifically, in Fig. 5.4, the F1 score is obtained in the individual ML test in APWD, whereas costs are calculated accordingly in POLC in the working procedure in Fig. 5.7. Steps in Section 5.2.1 for APWD can be revisited for POLC using costs to determine the winner. Therefore, the steps are skipped defining POLC and focus on the determination of the cost-based expert model.

Quantified coefficients for damage cost in DoS, Probe, $R2L$, and $U2R$ are illustrated in Fig. 5.6, which are introduced in [127]. D_i is used to denote the quantified damage

Table 5.12: F1 scores under APWD combining XGBoost, RF Adaboost and individual ML models under the NSL-KDD dataset

	APWD	Single model			Compare with single ML		
		RF	Adaboost	XGBoost	RF	Adaboost	XGBoost
DoS	0.889	0.877	0.400	0.889	1.3%	122.1%	0%
Norm	0.815	0.783	0.738	0.797	4.1%	10.5%	2.4%
Probe	0.760	0.749	0.383	0.746	1.5%	98.4%	1.8%
R2L	0.421	0.022	0.416	0.101	1853.7%	1.1%	318.6%
U2R	0.183	0.020	0.183	0.057	829.2%	0.0%	220.4%

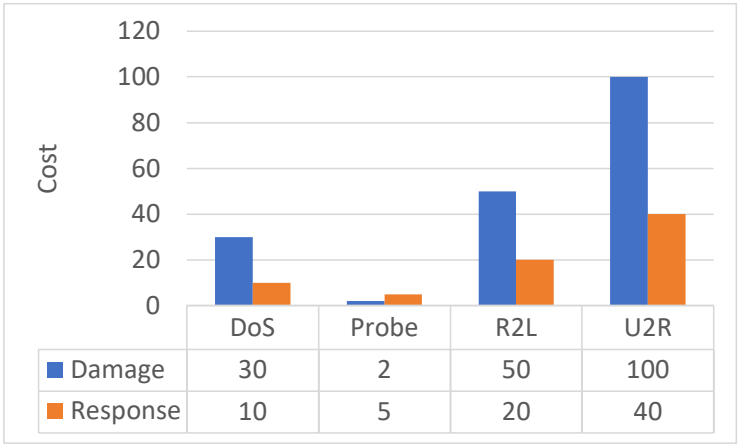


Figure 5.6: Attacks damage and response cost quantified value [127]

cost for each class. Moreover, Fig. 5.6 shows that $U2R$ results in more severe damage costs than R2L. An $R2L$ attack is performed by malicious users that gain unauthorized access to the victim's device. Meanwhile, in a $U2R$ attack, attackers illegally obtain the root privileges when legally connecting a local machine [219]. Although both $R2L$ and $U2R$ attacks are associated with improper access to user equipment for intrusion, it is obvious that $U2R$ results in more severe damages due to obtaining root privileges. Fig 5.6 presents the response cost denoted by R_i. In the NSL-KDD dataset, Normal class is included, resulting in 0 damage cost and 0 response cost, that does not display in Fig. 5.6.

POLC is presented to determine ML expertise for each cluster, considering both damage cost and response cost and aggregation module in POLC in the following sub-section.

MLx and MLy are obtained based on the maximum number of classes with the lowest cost and the second maximum number of classes with the lowest cost. The sets of classes

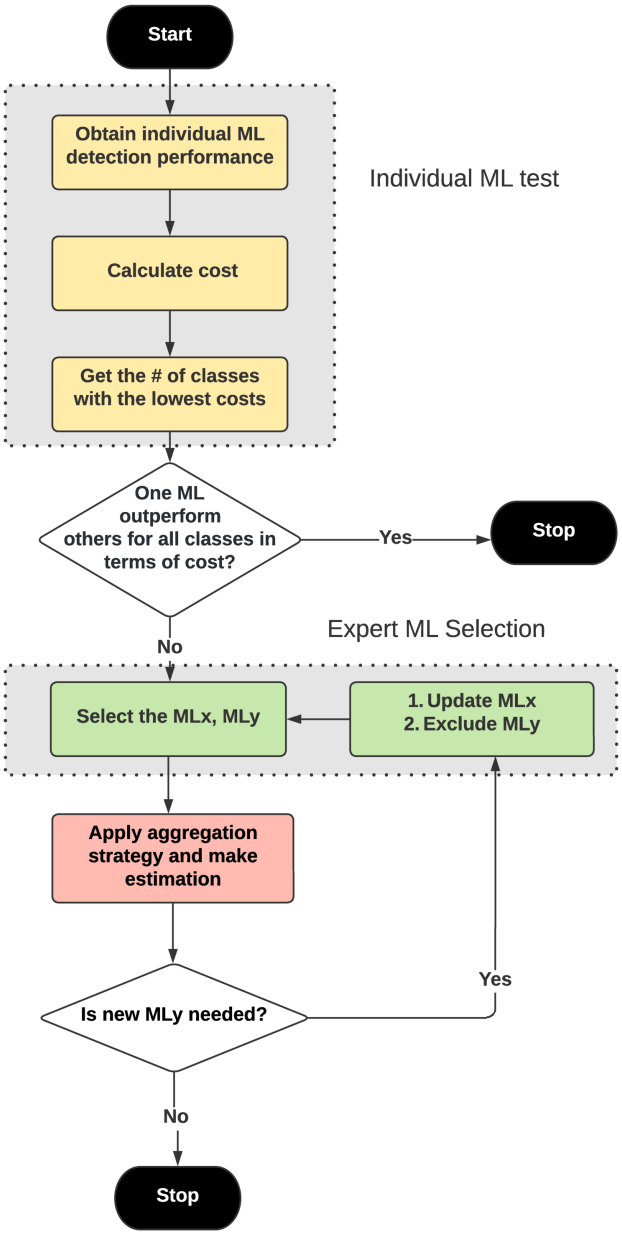


Figure 5.7: Flow of POLC

Table 5.13: Matrix coefficients for costs calculation under POLC

	DoS	Normal	Probe	R2L	U2R
DoS	-20	30	35	50	70
Normal	10	ϵ	5	20	40
Probe	12	2	3	22	42
R2L	60	50	55	-30	90
U2R	110	100	105	120	-60

for MLx and MLy with expertise are recorded and used in the aggregation module. Meanwhile, the method is described to calculate the system damage costs per attack, prediction cost per class, and overall cost.

POLC selects the base ML estimators according to the costs of their decisions. Combined response and damage cost coefficient $C_{rd}^{i,j}$ after the ML prediction is formulated in (5.1).

$$C_{rd}^{i,j} = \begin{cases} R_j + D_i, i \neq j \\ R_j - D_i, i = j, i \neq Normal \\ \epsilon, i = j = Normal \end{cases} \tag{5.1}$$

, where i denotes the true label and j stands for the prediction; R_j represents the response cost of class j and D_i is the damage cost of class i as shown in Fig. 5.6. Based on Fig. 5.6 and (5.1), combined damage and response cost coefficient is shown in Table 5.13. Cost calculation after an ML prediction is formulated in (5.2).

$$C_{i,j}^{ML} = C_{rd}^{i,j} \times CF_{i,j}^{ML} \tag{5.2}$$

In the equation above, j is the predicted class, and i is the true label. Furthermore, $C_{rd}^{i,j}$ denotes the combined damage and response cost coefficient whereas $CF_{i,j}^{ML}$ stands for the corresponding value in the confusion matrix after the prediction made by the ML model. Damage cost is determined by summing up the costs of all classes. Thus, (5.3) is applied for the total damage cost of a particular class (i.e., attack type).

$$TD_i^{ML} = \sum_j C_{i,j}^{ML} \tag{5.3}$$

In the equation above, TD_i^{ML} denotes the total damage cost for class i after the estimation of an ML model. Average costs for a class is presented as the total combined cost per sample as formulated in (5.4).

$$AC_i^{ML} = \frac{TD_i^{ML}}{Total\ samples\ of class\ i} \tag{5.4}$$

AC_i^{ML} is used to determine the expertise of an ML model for a particular class. The lower damage cost, the better the prediction performance for the class. An ML leading to the best performance for a particular class means the ML leads to the lowest prediction cost for that class. For instance, if $TD_i^{XGBoost}$ is smaller than TD_i^{DT} for the same class i, XGBoost is considered as the predictor for class i.

GAN based attack samples augmentation

NSL-KDD dataset contains significantly small amount $R2L$ and $U2R$ samples as shown in Table 5.1. An imbalanced dataset impedes the predictive capacity of classification methods. The imbalance issue can be solved via traditional oversampling techniques (e.g., SMOTE, ADASYN, and random oversampling [161]). However, these classical techniques are likely to create biased synthetic samples since they do not take into account the whole sample distribution [177]. A GAN model can cope with this phenomenon by generating samples that consider the entire data [177, 69]. In this study, a GAN model introduces the augmentation of $R2L$ and $U2R$ samples. Fig. 5.8 shows the originally trained dataset utilized to train the GAN, including its Discriminator and Generator neural networks. With the motivation to generate synthetic samples similar to original samples, the originally trained dataset is filtered. In the case of enlarging the $U2R$ sample size, Normal and $U2R$ data points in the originally trained dataset are picked up and sent to GAN. On the other hand, $R2L$ and Normal traffic is chosen as the input for GAN to augment the $R2L$ samples.

GANs start by taking noise samples as input so to generate adversarial samples. The training procedure of GAN is described below. The Generator utilizes noise instances to train itself, which transforms noise signals into meaningful output signals. The generator's objective is to generate samples mimicking actual data. Simultaneously, the discriminator seeks to differentiate real samples from generated data. The estimation error is sent to the Generator and the Discriminator networks using the backpropagation method outputs minority attack samples such as $R2L$ and $U2R$. The generated samples are injected into the original training set to obtain a new training set. The new training dataset is split for training and validation in the training procedure.

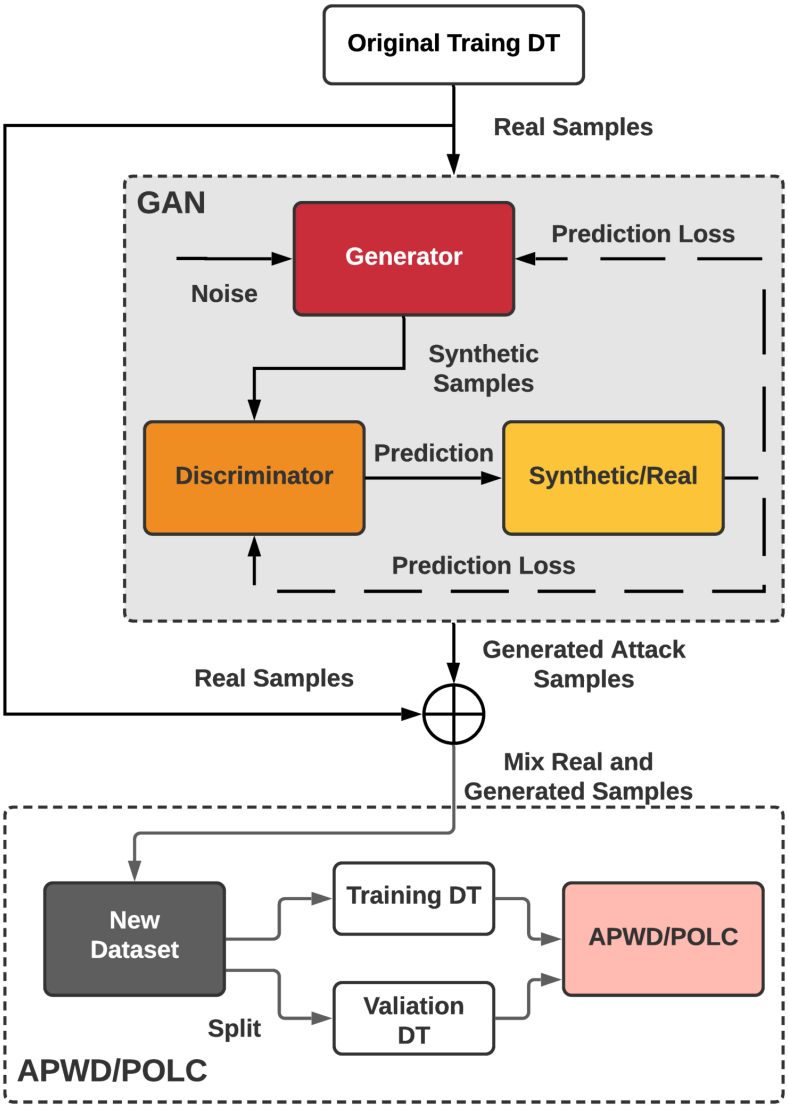


Figure 5.8: GAN integrated with APWD/POLC system

5.3.2 Performance evaluation under NSL-KDD dataset

Experimental results are presented for the APWD and POLC approaches under the NSL-KDD dataset where the numerical results under the designed method are compared to the classical ML algorithms, including XGBoost and DT.

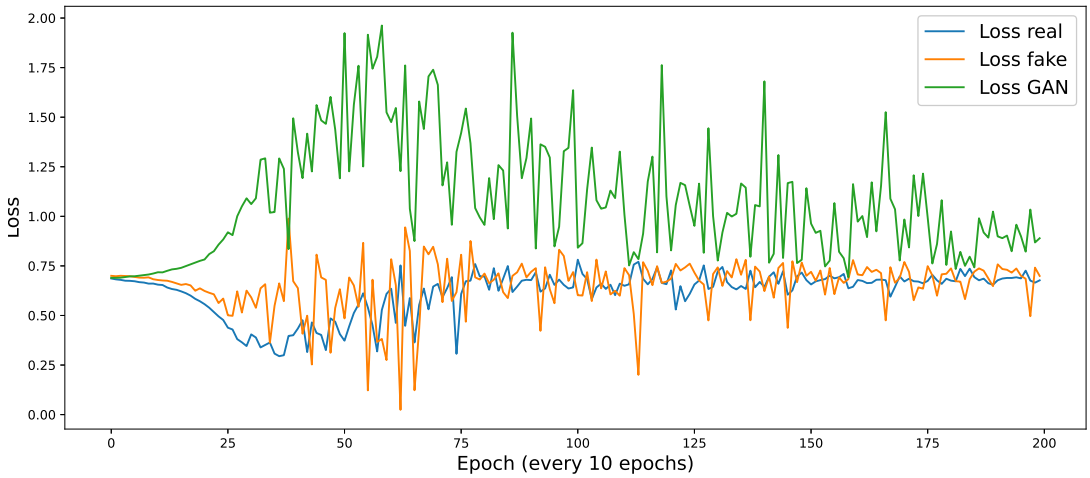
NSL-KDD Data augmentation

NSL-KDD dataset in Section 5.1 is chosen to verify the introduced method performance in identifying attacks, which is frequently used as a feasible benchmark for assessing different intrusion detection systems [263]. The dataset is composed of four distinct intrusion groups and normal traffic data. Table 5.1 lists NSL-KDD dataset distribution for each group. The NSL-KDD dataset is visualized in two dimensions in Fig. 5.1 via the t-SNE dimensionality reduction technique [154, 17].

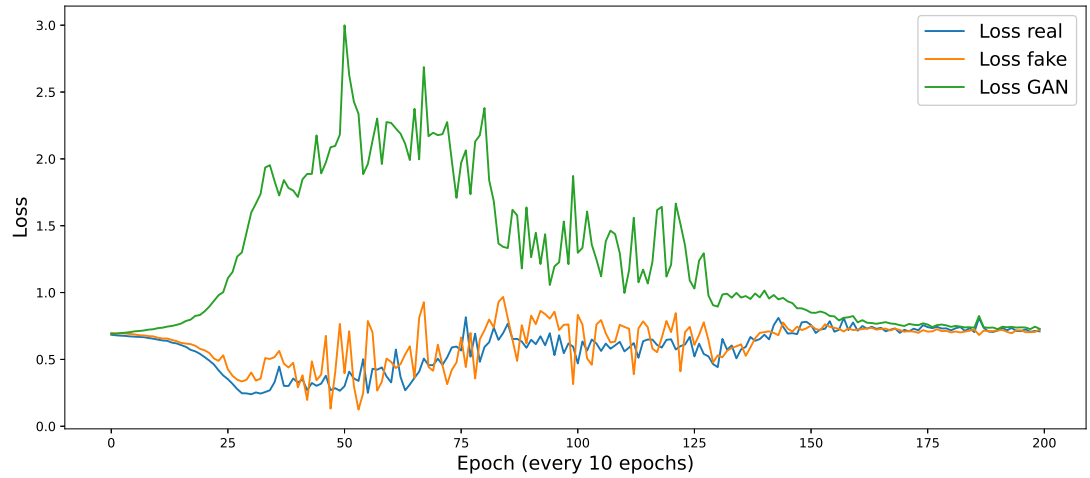
Majority of the NSL-KDD samples include normal traffic data whereas a limited number of attack samples are present, especially for $U2R$ and $R2L$ instances, as shown in Table 5.1. It is difficult for an ML algorithm to train itself well demonstrate promising performance with these rare samples. A GAN-based approach is introduced to address the rarity of $U2R$ and $R2L$ instances. In this work, $20,000$ $R2L$ and $51,000$ $U2R$ instances are generated separately. Fig. 5.9a and Fig. 5.9b illustrate the Discriminator's training loss using original samples in the training dataset (i.e., loss real), using created samples (i.e., loss fake), and the GAN model's training loss (i.e., loss GAN). Loss values are presented in Fig. 5.9a and Fig. 5.9b which are calculated using the binary cross-entropy loss function for each of the ten epochs. Here, $2,000$ epochs are configured during the GAN training procedure, with a steady reduction of train loss and convergence to a small value. The GAN introduces $20,000$ $R2L$ and $51,000$ $U2R$ by its Generator network to the original training dataset. The generated synthetic $U2R$ and $R2L$ samples are mixed with the original NSL-KDD dataset, which leads to two new training datasets. One new training dataset contains $20,995$ $R2L$ points and keeps original points for all other classes in NSL-KDD; another new training dataset includes $51,052$ $U2R$ samples and keeps original samples for all other classes in NSL-KDD. According to the system overview in Fig. 5.8, the new training dataset is split into validation and training sets for the APWD/POCL frameworks.

Performance evaluation of individual ML classifiers

Expert MLs determination for APWD Two ML classifiers (e.g., DT and XGBoost) are selected based on ML estimators in APWD. Under the APWD framework, individual



(a) Training loss during $R2L$ augmentation.



(b) Training loss during U2R augmentation

Figure 5.9: Training loss for GAN

MLs should be trained and verified via training and validation datasets. The expertise of an ML classifier for each class is determined by the validation results shown in Table 5.14 for DT and XGBoost. According to validation results for XGBoost, better performance is obtained with the original training dataset than using the two new training datasets. On the other hand, the performance of DT is more promising under the new training datasets when compared to its performance under the original training dataset. Thus, XGBoost uses the original dataset without any data augmentation for the minority classes. DT_R2L stands DT using the new dataset with $20,000$ synthetic $R2L$ samples for training, and DT_U2R denotes DT using the new dataset with $51,000$ synthetic $U2R$ samples for training. It can be observed that XGBoost outperforms DT_R2L and DT_U2R for DoS, Normal and Probe classes (with * in Table 5.14 in comparison to DT in $U2R$ and $R2L$. On the other hand, DT_U2R demonstrates the highest F1 score for $U2R$ with 0.9996, and DT_R2L illustrates the best performance for $R2L$ with 0.9963. Prediction results of the individual ML models are shown in Table 5.14, which are used to select expert ML algorithms in the APWD working procedure. Based on their F1 score rankings, the first (MLx) and second (MLy) candidates are XGBoost and DT_R2L. Upon the selection of these two classifiers, four classes (e.g., DoS, Normal, Probe and R2L) are covered in the first aggregation loop. $U2R$ is not covered in the first iteration so APWD proceeds with the second round of selection to add one more candidate, and DT_U2R is selected as MLy in the second round.

Expert MLs determination for POLC Confusion matrices under the verification datasets are shown in Table 5.15. According to Eq. 5.2 and confusion matrices, three cost tables are calculated in Table 5.16 (e.g., denoted by $C_{i,j}^{XGBoost}$, $C_{i,j}^{DT\_R2L}$, and $C_{i,j}^{DT\_U2R}$). Table 5.17 is obtained to demonstrate the average damage cost per class for each ML model, which is used to determine the cost performance of the ML models for specific class(es). The less average damage cost means better performance for a class, which is marked by $*$ in Table 5.17. Table 5.17 shows that XGBoost demonstrates low-cost performance for Normal, DoS, and Probe traffic patterns; DT_U2R results in the least damage cost for $U2R$; and DT_R2L results in the minimum damage cost for $R2L$ traffic. APWD and POLC select the same expert MLs for each class as presented in Table 5.14 and Table 5.17. It means the same MLx (XGboost) and MLy (DT_R2L and DT_U2R) are applied in the aggregation procedure in the subsequent section.

APWD/POLC Prediction

Based on the previous introduction of single ML estimator performance and expertise selection, XGBoost and DT_R2L are chosen as MLx and MLy for the first time aggregation.

Table 5.14: F1 score by DT and XGBoost with different datasets. Entry with a * denotes the highest rank of F1 score for the corresponding class.

	DT_R2L	DT_U2R	XGBoost
DoS	0.9996	0.9995	0.9999*
Normal	0.9974	0.9967	0.9991*
Probe	0.9925	0.9913	0.9979*
R2L	0.9963*	0.9134	0.9859
U2R	0.5882	0.9996*	0.8000

The aggregation procedure in Fig. 5.4 checks the prediction results for each sample in the test dataset; the aggregation procedure follows DT_R2L estimation when DT_R2L predicts this sample $R2L$; otherwise, the decision follows XGBoost prediction of result. So far, four groups (e.g., DoS, Normal, Probe, and $R2L$) are covered in the first aggregation since XGBoost shows expertise in DoS, Normal, Probe and DT_R2L demonstrates expertise in $R2L$. Meanwhile, the first aggregation results are updated as a new MLx. The next step is to find a new MLy (DT_U2R), combined with the new MLx for Normal class to make a wise prediction. The aggregation procedure follows the first time aggregation but with new MLx and new MLy as shown in Fig. 5.4. APWD and POLC merge three ML-based estimators, and performance is illustrated in Table 5.18. The overall system performance is raised to 0.7862, which is the best performance in comparison to 0.7717 under XGBoost, 0.7726 under DT with the dataset with $R2L$ synthetic samples, and 0.7839 under the dataset with $U2R$ synthetic samples. Table 5.19 compares F1 score among traditional ML algorithms and APWD/POLC frameworks. APWD improves the F1 scores for most attack types when compared to DT and XGBoost, especially for $U2R$ and $R2L$. For instance, APWD boosts $R2L$ F1 score by 192% and $U2R$ by 127% compared to XGBoost. Meanwhile, $R2L$ F1 score is improved by 60% under the APWD framework for DT_R2L. APWD leads to a side effect for some classes. For example, there is 10% reduction for Probe and a slight decline (1%) for U2R.

The authors in [62] present detection performance using several well-known ML models under the NSL-KDD dataset that are compared with the results in this thesis. Fig. 5.10 demonstrates performance comparison with RF, SVM, and LSTM in [62]. For DoS attack, APWD/POLC demonstrates 0.8261 recall, showing a marginal decrease, compared to RF 0.8291 in [62]. Other than DoS, APWD/POLC demonstrates a higher recall than RF, SVM, and LSTM in [62], especially for $R2L$ and $U2R$ attacks. The proposed framework APWD/POLC contributes the highest detection rate for most attack types. For instance, APWD/POLC results in a detection rate of 0.1725 for $R2L$, which significantly increases

Table 5.15: DT and XGBoost prediction confusion matrices under different verification datasets

		Prediction Label				
True Label	**XGBoost**	DoS	Normal	Probe	R2L	U2R
	DoS	9115	1	0	0	0
	Normal	1	13501	1	1	2
	Probe	0	9	2336	0	0
	R2L	0	5	0	210	0
	U2R	0	3	0	0	10
	DT_R2L	DoS	Normal	Probe	R2L	U2R
	DoS	9153	0	1	0	0
	Normal	3	13491	13	13	2
	Probe	2	18	2259	0	0
	R2L	1	17	0	4212	0
	U2R	0	5	0	0	5
	DT_U2R	DoS	Normal	Probe	R2L	U2R
	DoS	9139	5	0	0	0
	Normal	4	13352	19	13	2
	Probe	0	22	2349	0	0
	R2L	0	19	0	174	0
	U2R	1	5	0	1	10000

Table 5.16: Total costs of DT and XGBoost under different verification datasets. Rows and columns indicate true labels and predicted labels, respectively.

XGBoost	DoS	Normal	Probe	R2L	U2R
DoS	-182300	30	0	0	0
Normal	10	135.01	5	20	80
Probe	0	18	7008	0	0
R2L	0	250	0	-6300	0
U2R	0	300	0	0	-600
DT_R2L	DoS	Normal	Probe	R2L	U2R
DoS	-183060	0	35	0	0
Normal	30	134.91	65	260	80
Probe	24	36	6777	0	0
R2L	60	850	0	-126360	0
U2R	0	500	0	0	-300
DT_U2R	DoS	Normal	Probe	R2L	U2R
DoS	-182780	150	0	0	0
Normal	40	133.52	95	260	80
Probe	0	44	7047	0	0
R2L	0	950	0	-5220	0
U2R	110	500	0	120	-600000

Table 5.17: Average costs under POLC. $\epsilon = 0.01$. * denotes the minimum cost for the corresponding class (e.g., column header) by a specific ML model (e.g., row header).

	DoS	Normal	Probe	R2L	U2R
DT_U2R	-19.97	0.13	3.01	-25.08	-59.95*
DT_R2L	-19.99	0.11	3.02	-29.81*	-22.00
XGBoost	-20.00*	0.05*	2.99*	-29.21	-40.00

Table 5.18: Test performance of the individual MLs and APWD/POLC. Prec: Precision

XGBoost	Prec	Recall	F1	Accuracy
DoS	0.962	0.826	0.889	
Normal	0.675	0.971	0.797	
Probe	0.822	0.683	0.746	0.7717
R2L	0.967	0.053	0.101	
U2R	0.600	0.030	0.057	
DT_R2L	Prec	Recall	F1	Accuracy
DoS	0.956	0.809	0.877	
Normal	0.686	0.929	0.789	
Probe	0.734	0.782	0.757	0.7726
R2L	0.980	0.164	0.281	
U2R	0.517	0.075	0.131	
DT_U2R	Prec	Recall	F1	Accuracy
DoS	0.9552	0.8113	0.8774	
Normal	0.6876	0.9691	0.8045	
Probe	0.8694	0.7922	0.8290	0.7839
R2L	0.9686	0.1009	0.1828	
U2R	0.5000	0.0750	0.1304	
APWD/POLC	Prec	Recall	F1	Accuracy
DoS	0.9618	0.8261	0.8888	
Normal	0.6922	0.9700	0.8079	
Probe	0.8217	0.6832	0.7461	0.7862
R2L	0.9794	0.1725	0.2933	
U2R	0.4839	0.0750	0.1299	

Table 5.19: Performance (F1 score) of APWD/POLC and base classifiers (XGBoost, DT). DT_U2R:Decision Tree model trained with synthetic U2R samples; DT_R2L:Decision Tree model trained with synthetic R2L samples

	Proposed	Base Estimator			Compare With Base Estimator		
		DT_U2R	DT_R2L	XGBoost	DT_U2R	DT_R2L	XGBboost
DoS	0.889	0.877	0.877	0.889	1%	1%	0%
Normal	0.808	0.804	0.789	0.797	0%	2%	1%
Probe	0.746	0.829	0.757	0.746	-10%	-1%	0%
R2L	0.293	0.183	0.281	0.101	60%	4%	192%
U2R	0.130	0.130	0.131	0.057	0%	-1%	127%

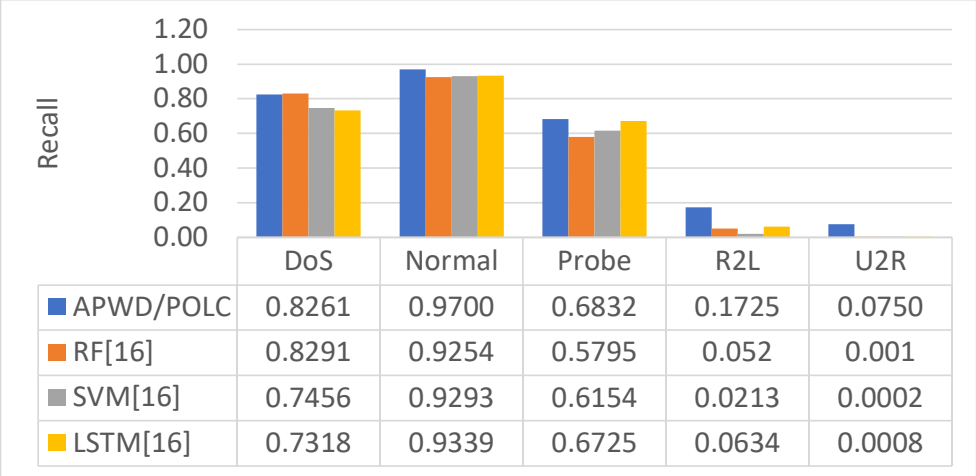


Figure 5.10: Recall comparison between APWD/POLC, RF, SVM and LSTM

the detection rate of 0.052 by RF, 0.0213 by SVM, and 0.0634 by $LSTM$. Meanwhile, the authors in [62] apply deep neural networks such as DBN and CNN to build intrusion detection systems. These deep neural networks are not compared with our work since this work focuses on leveraging off-the-shelf and relatively easy-to-train machine learning algorithms rather than deep networks.

5.3.3 Performance evaluation under the SCVIC-APT-2021 dataset

APWD approach is also evaluated under the SCVIC-APT-2021 dataset [147]. However, POLC is not evaluated under SCVIC-APT-2021 dataset as presented in Section 5.1. Since

Table 5.20: RF, XGBoost, Adaboot performance comparison, including precision, recall and F1 score, using private dataset. Overall accuracy for RF, XGBoost and Adaboost is 0.996, 0.995, and 0.522, respectively. DE=Data Exfiltration, IC=InitialCompromise, LM=LateralMovement, Nor=Normal, Piv.=Pivoting, Rec.=Reconnaissance

	ML	DE	IC	LM	Nor	Piv.	Rec.
Pre	RF	0.737	0.921	0.757	1.000	0.779	0.774
	Xgb	0.395	0.878	0.937	1.000	0.611	0.776
	Ada	0.185	0.000	0.332	0.998	0.729	0.678
Rec	RF	0.189	0.753	0.810	1.000	0.939	0.725
	Xgb	0.203	0.844	0.831	0.999	0.994	0.359
	Ada	0.378	0.169	0.493	0.521	0.667	0.705
F1	RF	0.301*	0.829	0.782	1.000*	0.851*	0.749*
	Xgb	0.268	0.861*	0.881*	1.000	0.757	0.490
	Ada	0.249	0.001	0.397	0.684	0.697	0.691

there is no evidence to quantize damage cost and response cost for attacks in the SCVIC-APT-2021 dataset, costs can not be calculated in the POLC approach.

This study selects three base ML estimators (e.g., RF [188], XGBoost [44] and Adaboost [207]) to evaluate APWD framework performance under the SCVIC-APT-2021 dataset. Test results of the individual models are presented in Table 5.20. To select suitable candidates (e.g., MLx and MLy) in the APWD model, Table 5.20 presents F1 score comparison for the three models. RF outperforms XGBoost and Adaboost in classifying four classes (i.e., DataExfiltration, normal traffic, Pivoting and Reconnaissance). XGBoost performs better in identifying three classes: InitialCompromise, LateralMovement and Normal traffic. Adaboost does not obtain outperform the other algorithms for any of the traffic classes in terms of F1 scores; therefore, Adaboost is not a candidate for the APWD framework. Hence, the first candidate MLx is RF, with four F1 scores (of different classes) ranking first, whereas the second candidate MLy is XGBoost, with three F1 scores (of different classes) ranking first. Meanwhile, all classes have been covered by the first and second candidates, so there is no need to seek further ML models for APWD. Based on the previous analysis, key parameters used in the APWD system aggregation procedure are demonstrated in Table 5.21, and the general descriptions regarding the outputs of ML algorithms and tracking of the F1 score ranking of ML algorithms are given in Table 5.5. The aggregation process runs according to the APWD aggregation rule illustrated in Fig. 5.4.

Table 5.21: F1 score performance of RF, XGBoost models under the SCVIC-APT-2021 dataset, and inputs to APWD selection

		Machine Learning Model Expertise For Each Class	
		RF	XGBoost
Class prediction	DataExfiltration	0.301	0.268
	InitialCompromise	0.829	0.861
	LateralMovement	0.782	0.881
	NormalTraffic	1.000	1.000
	Pivoting	0.851	0.757
	Reconnaissance	0.749	0.490
Exp_max_num		4	2
Set of classes with highest F1 score in ML		SPx = { DataExfiltration, NormalTraffic, Pivoting, Reconnaissance}	Spy = { InitialCompromise, LateralMovement }

APWD system makes an estimation for each instance in the test dataset, and the detection performance is shown in Fig. 5.11. APWD improves the overall accuracy marginally when compared to RF and XGBoost. The results in Table 5.22 demonstrate that the proposed framework APWD significantly improves attack detection performance for all classes when compared with RF. For instance, LateralMovement F1 scores are 0.782 and 0.829 under RF and APWD, respectively. Thus, approximately 6% improvement is introduced via APWD. Furthermore, APWD contributes higher F1 scores for most of the classes than XGBoost. Specifically, the detection performance (F1 score) of Reconnaissance is increased from 0.49 to 0.76, with an improvement of 55.1%. However, there is a 2.6% drop in terms of the F1 score for InitialCompromise and a 5.8% reduction for LateralMovement under APWD. The root cause of the performance decrease is the detection accuracy not being 100% for all classes. The IntialCompromise and LateralMovement decisions of APWD rely on XGBoost whereas its decisions for the other four classes rely on RF. RF leads to false-positive prediction in InitialCompromise, and LateralMovement when predicting the other four classes, which decreases the prediction performance for some classes.

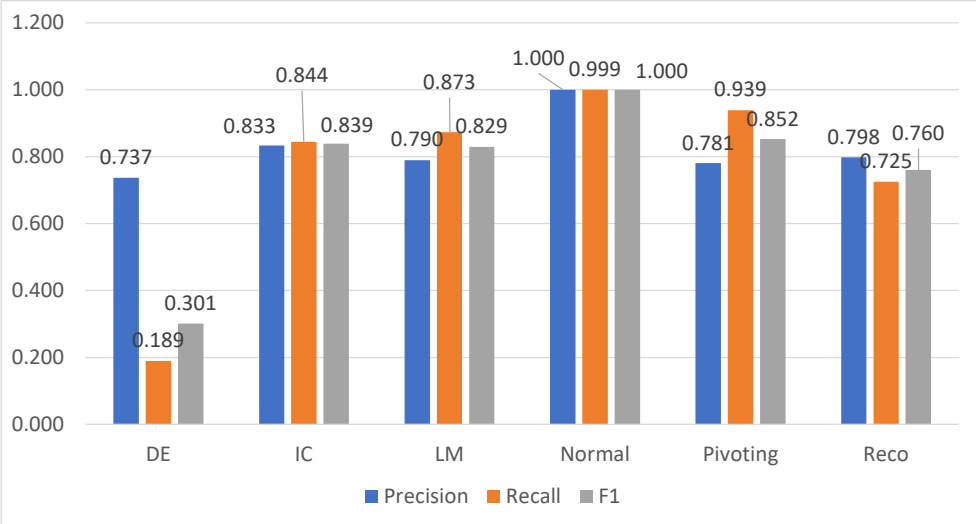


Figure 5.11: APWD performance combining XGBoost and RF under the SCVIC-APT-2021 dataset with overall accuracy 0.996

5.3.4 Result analysis

DT in POLC leads to the lowest cost for its $U2R$ (-59.95 units) and $R2L$ (-29.81 units) decisions, whereas DT in APWD demonstrates the highest F1 score in $U2R$ (0.9996) and $R2L$ (0.9963). Both the APWD and POLC frameworks improve accuracy, from 0.7717 (under XGBoost) to 0.7826 by APWD/POLC. Moreover, the proposed models enhance detection performance for most attack groups. Specifically, the APWD / POLC model improves the $R2L$ F1 score by up to 192%. Besides, the SCVIC-APT-2021 dataset is also considered to test and present the generalization of the APWD framework. APWD taking RF and XGBoost as base estimators to determine expert ML for each type of attack. APWD performs the highest F1 score in DataExfiltration, Pivoting, and Reconnaissance. Specifically, APWD improves Reconnaissance F1 score to 0.76, compared with 0.49 by XGBoost, with a 54.9% increment. However, the APWD/POLC system diminishes performance for some classes. For instance, APWD leads to Probe F1 score decreasing 1-10% and $U2R$ F1 score reducing 1%, compared to DT using NSL-KDD dataset.

Table 5.22: F1 scores in APWD combing XGBoost and RF vs single ML models under the SCVIC-APT-2021 dataset

	APWD	Single model		Comparing with single model	
		RF	XGBoost	RF	XGBoost
DataExfiltration	0.301	0.301	0.268	0.0%	12.4%
InitialCompromise	0.839	0.829	0.861	1.2%	-2.6%
LateralMovement	0.829	0.782	0.881	6.0%	-5.8%
NormalTraffic	1.000	1.000	1.000	0.0%	0.0%
Pivoting	0.852	0.851	0.757	0.1%	12.6%
Reconnaissance	0.760	0.749	0.490	1.5%	54.9%

5.4 Host-Based Network Intrusion Detection Under Feature Flattening and Cascade Machine Learning

The advances in information and communication technologies have increased the importance of security solutions for networked systems [48, 271]. With more access points, attack surfaces become widespread [64], leading to more vulnerable targets against cyber-attacks even if defense mechanisms are in place [171]. NIDSs have been widely investigated and implemented to protect network systems [225]. ML-based NIDS has been proven to enable NIDS to detect popular attacks and zero-day attacks which has also been widely investigated in the context of defensive and proactive / adversarial ML [146]. An example of a ML-driven NIDS is an XGBoost-DNN integrated NIDS [59], which has been shown to demonstrate decent performance.

NIDS builds on captured global network traffic by analyzing data from a single packet to identify suspicious activities [87]. However, it is not easy in some scenarios to monitor the entire network traffic, such as encrypted traffic, or without monitoring authentication. Meanwhile, NIDS calls for improved solutions to identify particular malicious activities, such as APT, which generally utilize malicious files attached to various applications [166]. Host-based intrusion detection aims to detect attacks on hosts (e.g., servers) by evaluating resources in a host, including logs, files, folders. HIDS is beneficial as it offers fine-grained solution to detect anomalous patterns internally [150]. Therefore, the role of HIDS in the detection of ATP is crucial. The authors in [165] show that APT is detectable via HIDS in an efficient manner. Moreover, HIDS is meritorious as it detects anomalies without

analyzing or monitoring the network traffic. The authors in [202] introduce an HIDS use case to secure Android mobile equipment. An intrusion detection system is proposed, integrating both HIDS and NIDS, that considers real network traffic and records saved in host devices as illustrated in Fig. 5.12.

The CICIDS 2018 dataset and NDSec-1 dataset are public datasets used in intrusion detection schemes [220, 174]. The CICIDS 2018 dataset and NDSec-1 contain host-based information, including logs for events and messages. However, very few studies apply host information in intrusion detection [42, 28]. It is worth noting host-based contents are stored as a string, which is transformed to a numeric array by Bert [226]. It means event data and message data that are saved in the host are transformed separately to obtain host-based features. Fig. 5.13 illustrates the structure of network-based and host-based features in CICIDS 2018 and NDSec-1 datasets. Message text and event text information are transformed into two dimensional numerical matrices separately by Bert. Specifically, Bert transforms event text into a $n * 8$ matrix and message text into a $m * 768$ matrix. Depending on the dataset, n and m represent different dimensions. Similarly, flow feature / network feature dimension is represented by a $1 * f$ matrix, where f is determined by the number of network features in the dataset. Thus, such transformation enables machine learning algorithms to utilize flow features and host features to train themselves which get benefits from network intrusion detection system and host-based intrusion detection system [148].

On the other hand, one of the motivation behind this work is that intrusion datasets contain huge amounts of samples. For instance, there are about 1 million samples in CICIDS 2018, as shown in Table 5.3. Normally, benign traffic dominates dataset and attack samples are minority since limited number of attack points can be hidden properly (except denial-of-service related attacks). Consequently, large amount benign samples increases the complexity of the intrusion detection system due to increased training overhead of the ML models. This work leverages network-based and host-based intrusion detection frameworks concurrently to inherit the benefits of both systems, and proposes a two-stage framework to boost intrusion detection accuracy and minimize false alarm probability. In order to reduce detection system complexity and maintain detection performance, this work introduces a novel framework cascading two level detection strategy, which comprises of a binary classifier and multi-classifier. The binary classifier aims to discriminate benign and attack instances (all attacks share the same label) whereas the multi-class classifier focuses on characterizing the attack instances after the benign instances have been filtered out by the binary classifier.

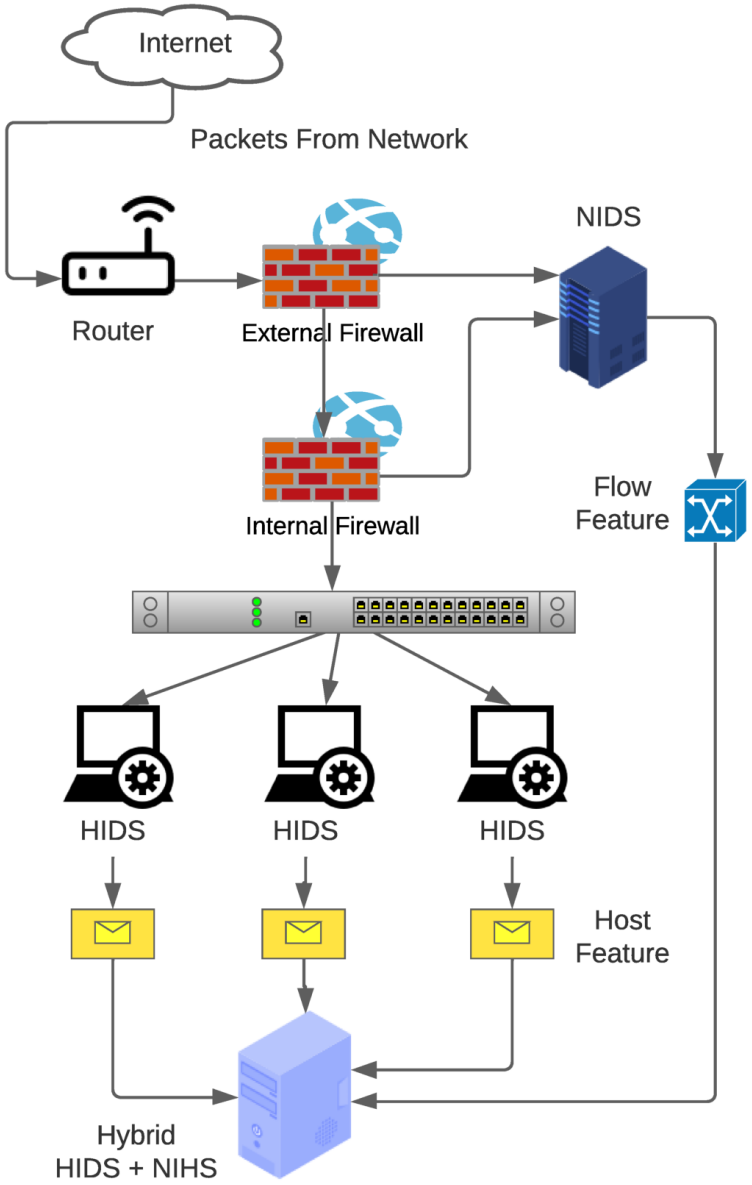


Figure 5.12: An example of hybrid HIDS and NIDS system

5.4.1 Hybrid network features and host features based on intrusion detection method

This section explains the framework to combine network and host features for machine learning-integrated intrusion detection. It utilizes network traffic and records in host devices as shown in Fig. 5.12.

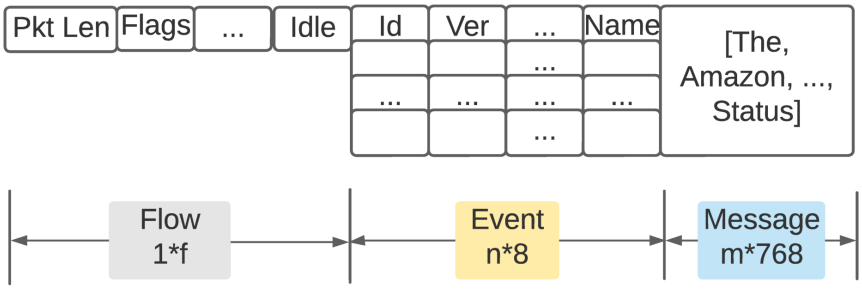


Figure 5.13: Flow features and host features after transforming. Initially presented in [148].

In this work, the proposed strategy is adopt [148], and Bert is employed for word embedding to extract host features, a transformer architecture trained to obtain language representations. Event data and message data are derived separately which both have two dimensions as presented in Fig. 5.13 (as presented in [148]). Flow features, also called network-based features, are the major features in NIDS, especially in ML-based detection systems [59, 9, 101]. In order to utilize host-based features, a two-dimensional feature data is flattened into a one-dimensional vector. As a result, host-based features can be conveniently used as network-based features by machine learning and deep learning models. Feature flattening is used in Convolutional Neural Networks (CNN) due to the lack of support for multidimensional data in densely-connected layers of a CNN [6, 9]. Feature flattening is a feasible approach to adjust multidimensional data to vectorized features, especially to be used by ML algorithms.

Feature reduction

Feature flattening makes ML algorithms to use host features easily. However, it leads to feature flooding issue. Table 5.23 demonstrates the number of event features and message features dimensions. Specifically, in the CICIDS 2018 dataset, event features are extracted from a 28∗8 matrix ($n = 28$). Meanwhile, message feature dimension is 100∗768 ($m = 100$).

Table 5.23: Original dimension of CICIDS 2018 and NDSec-1 for flow, event, and message features

	Flow	Event	Message
CICIDS	1*132(f=132)	28*8(n=28)	100*768(m=100)
NDSec-1	1*63(f=63)	2182*8(n=2182)	512*768(n=512)

After flattening the event and message features, $1*226$ event features and $1*76800$ message features are obtained for every sample. Thus, a sample has a maximum of $77,156$ features in total. With more than 1 million samples in the training dataset, it is not easy to apply a total of $77,156$ features under the current computer computation capability. For the NDSec-1 dataset, the number of flow features is 63. Meanwhile, event features are extracted in a $2182 * 8$ matrix which denotes n as 2182. Moreover, the dimension of message features is $512 * 768$ ($m = 100$). Thus, a sample in the NDSec-1 dataset consists of a maximum number of features up to $410,735$ when mixing flattened host features and network features. With overwhelming number of features to be processed by an ML model, the NDSec-1 dataset confronts the same dimensionality challenge as the CICIDS 2018 dataset.

With this in mind, this work reduces the message feature matrix and event feature matrix before flattening them into a vector. For CICIDS 2018 dataset, reduction of message features need to be considered since message features constitutes the majority (e.g., 76,800) among all feature types. Shortening the message feature matrix from $100*768$ to a smaller matrix such as $10*10$ reduces message features directly. On the other hand, it is critical to reduce both event features (i.e., $2182*8$) and host features (i.e., $512*768$) for the NDSec-1 dataset before flattening them into a vector array due to a large number of features for event and host. Since an appropriate number of event features and message features is required before shortening, a method needs to be designed to select a small matrix out of the original large matrix. This work applies the random selection method to determine rows and columns prior to feature flattening. The random selection method aims to ensure the selected message matrix is representative of the large one and less likely to be subject to bias [40]. For instance, when message features in CICIDS 2018 are aimed to be reduced from $100*768$ to $10*15$, 10 is randomly selected among range 1 to 100 and 15 is randomly selected from range 1 to 768. It is worth noting that the average of several rounds of test results is measured to reduce the impact of fluctuations across samples and to take all features into account.

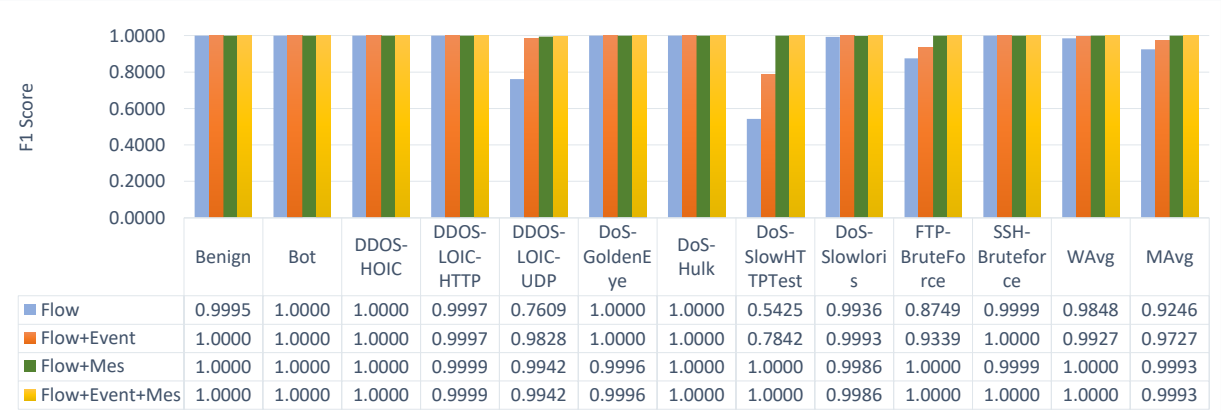
	Benign	Bot	DDOS-HOIC	DDOS-LOIC-HTTP	DDOS-LOIC-UDP	DoS-GoldenEye	DoS-Hulk	DoS-SlowHTTPTest	DoS-Slowloris	FTP-BruteForce	SSH-Bruteforce	WAvg	MAvg
10x80	1.0000	1.0000	1.0000	0.9999	0.9942	0.9983	1.0000	0.9999	0.9964	1.0000	1.0000	1.0000	0.9990
15x50	1.0000	1.0000	1.0000	0.9999	0.9942	0.9996	1.0000	1.0000	0.9986	1.0000	1.0000	1.0000	0.9993
20x40	1.0000	1.0000	1.0000	0.9999	0.9942	0.9996	1.0000	1.0000	0.9986	1.0000	1.0000	1.0000	0.9993
30x25	1.0000	1.0000	1.0000	0.9999	0.9942	0.9996	1.0000	1.0000	0.9986	1.0000	1.0000	1.0000	0.9993
40x20	1.0000	1.0000	1.0000	0.9999	0.9942	0.9996	1.0000	1.0000	0.9986	1.0000	1.0000	1.0000	0.9993
50x15	1.0000	1.0000	1.0000	0.9999	0.9942	0.9996	1.0000	1.0000	0.9986	1.0000	1.0000	1.0000	0.9993
80x10	1.0000	1.0000	1.0000	0.9999	0.9942	0.9996	1.0000	1.0000	0.9986	1.0000	1.0000	1.0000	0.9993

Figure 5.14: Performance comparison under various dimensions of message feature (host feature)

Performance under CICIDS 2018 dataset

As stated before, the message features matrix should be reduced before flattening. The original dimension is $100 * 768$, and our current computational capability can handle 800 host features. Detection performance with various matrices of message features is shown in Fig. 5.14 using XGBoost under the CICIDS 2018 dataset, which contains 10 types of attacks and excludes 4 types of attack having limited numbers (e.g., Brute Force-Web, Brute Force-XSS, Infiltration, and SQL Injection). The first 10 attack scenarios are presented in Table 5.3 for the training and test sets. The last 4 types of attack samples, as shown in Table 5.3, are limited in total (e.g., Brute Force-Web 42, Brute Force-XSS 16, Infiltration 29, and SQL Injection 15). Due to insufficient samples in CICIDS 2018 dataset, it is challenging to train machine learning and deep learning algorithms. Compared with the massive number of other attack scenarios, the limited samples are potentially considered noise in the training procedure and perform a lower detection performance. Therefore, the four classes are eliminated in most test cases except in the cascade machine learning approach. The results in Fig. 5.14 have been obtained under the same flow features (132) and the same event features (226). The message matrix is reduced to $10 * 80$, $15 * 50$, and $20 * 40$. It is observed that the performance is almost the same under different message

matrices. Message feature matrix $15 * 50$ is selected as a representative in the subsequent tests.



	Benign	Bot	DDOS-HOIC	DDOS-LOIC-HTTP	DDOS-LOIC-UDP	DoS-GoldenEye	DoS-Hulk	DoS-SlowHTTPTest	DoS-Slowloris	FTP-BruteForce	SSH-Bruteforce	WAvg	MAvg
Flow	0.9995	1.0000	1.0000	0.9997	0.7609	1.0000	1.0000	0.5425	0.9936	0.8749	0.9999	0.9848	0.9246
Flow+Event	1.0000	1.0000	1.0000	0.9997	0.9828	1.0000	1.0000	0.7842	0.9993	0.9339	1.0000	0.9927	0.9727
Flow+Mes	1.0000	1.0000	1.0000	0.9999	0.9942	0.9996	1.0000	1.0000	0.9986	1.0000	0.9999	1.0000	0.9993
Flow+Event+Mes	1.0000	1.0000	1.0000	0.9999	0.9942	0.9996	1.0000	1.0000	0.9986	1.0000	1.0000	1.0000	0.9993

Figure 5.15: NIDS (flow-based) and hybrid NIDS and HIDS (flow, event, and message-based) performance comparison under CICIDS 2018. Mes:message.

Fig. 5.15 demonstrates XGBoost test performance (i.e., F1 score) with flow features only, a combination of flow and message features, and a mixture of flow, message, and event features. The results show that integrating message features and event features improves the detection performance of all classes that cannot be fully detected. Particularly for the DDOS-LOIC-UDP attack, F1 score increases from 0.7609 (flow features only) to 0.9828 (with event features) and 0.9942 (by integrating message and event features). The largest improvement is achieved for the detection of DoS-SlowHTTPTest attack, which boosts from 0.5425 to 1.0000. Moreover, the overall performance is increased remarkably. For instance, the macro F1 score is increased up to 0.9993 with the event and message features when compared to 0.9246 with flow features only. The host features, be it event or message, clearly improve the attack detection accuracy. Moreover, message features show further advantages to boost the intrusion detection performance when compared to the event features. When flow features and message features are combined, 0.9993 macro F1 score is achieved while integrating flow features and event features lead to a macro F1 score of 0.9727.

Performance under NDSec-1 dataset

A similar rule is followed to apply host features for the NDSec-1 dataset by flattening two-dimensional host features into a vector for machine learning algorithms. As introduced in

Section 5.1.4, event features are stored in a $2182 * 8$ matrix whereas the message features are stored in a $512 * 768$ matrix. After flattening, a huge vector is obtained with $393,216$ message features for one sample. Therefore, dimensionality reduction is crucial before flattening by selecting a small matrix similar to the CICIDS 2018 dataset. According to previous experience under CICIDS 2018, dimensions of the message feature matrix impact he intrusion detection performance marginally, as shown in Fig. 5.14. The event feature matrix is selected as $500 * 8$, and the message feature matrix is $100 * 768$ for the NDSec-1 dataset. If a small matrix such as $10 * 10$ is chosen instead of the original $100 * 768$ event feature matrix, majority of the event features are excluded. One may expect this to reduce the effectiveness of the detection performance. However, with a smaller matrix of event and message features, the number of host features is also large; up to 80800 (4000+76800). Therefore, Principle Component Analysis (PCA) is employed to reduce dimensions further for the mix of the flow, event, and message features. Fig. 5.16 presents a comparison of performance results (F1 score) using PCA to reduce to different dimensions. This figure shows that the highest F1 score is achieved when PCA reduces the features to 10, with 0.91 macro F1 score. Thus, in the following tests, PCA is applied to reduce the features to 10. Fig. 5.17 presents the attacks detection performance (F1 score) with flow and host features. The detection system using the composition of flow features, event features, and message features contributes the highest overall performance in macro average F1 score (MAvg) with 0.8913 and a weighted average F1 score (WAvg) of 0.9964. Meanwhile, with a mixture of flow, event, and host features, the detection system demonstrates the best performance for DoS, Normal, and Probe. Specifically, MAvg is at 0.8913 when compared to the flow only case 0.8595, combined flow and event features (0.8591) and the combination of flow and message features (0.8906). When flow and event features are combined, the best performance for Botnet and Exploit is 0.9217 and 0.7933, respectively. Moreover, the flow and message combination case performs the best under Malware (0.9091) and Misc (0.7762). Thus, host features are helpful to boost attack detection performance in terms of overall performance and most individual classes when compared to the case where only network features are used. However, introducing host features lead to performance reduction in the detection of Bruteforce and Webattack. For instance, Bruteforce F1 score reduces slightly from 0.9985 (flow only) to 0.9981 (flow and event), and Webattack reduces from 0.8571 (flow only) to 0.7974 (flow, event, and message together).

5.4.2 Cascade Machine Learning Approach

In the dataset CICIDS 2018, benign samples form the majority, with $308,375$ in the training dataset and $152,172$ in the test dataset, contributing to approximately 50% in the training

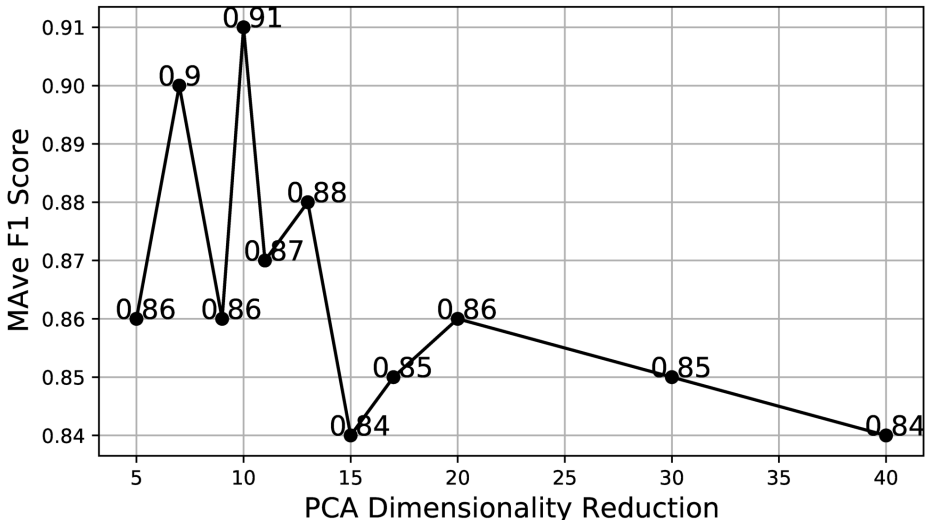
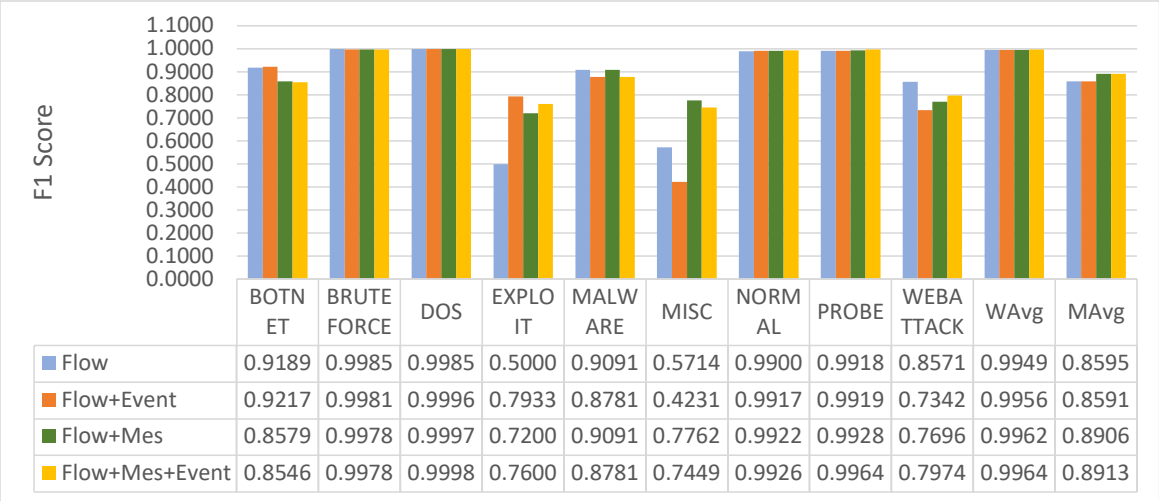


Figure 5.16: Comparison of the impact of various dimensions (reduced by PCA) under the NDSec-1 in terms of macro average F1 score

and test dataset as illustrated in Table 5.3. Moreover, according to the previous test results in Section 5.4.1, it can be observed that benign samples can be detected at an accuracy level that is close to 100%. Thus, attack points do not affect the detection performance of benign samples. If benign samples could be eliminated, the computational complexity of the intrusion detection system would be reduced. Meanwhile, it is possible to improve intrusion detection (per attack type) performance when the entire dataset complexity is reduced after filtering out half of the samples in the dataset. Therefore, a cascade Machine Learning (ML) framework is proposed as shown in Fig. 5.19, which integrates a binary classifier $ML1$ and a multi-class classifier $ML2$. The training and test procedures are conducted sequentially. The training dataset is used to train $ML1$ in the first stage. In this case, the training dataset is labeled with two classes, including 0 and 1 to represent benign and attacks, respectively. All attack classes share one label as attack, and benign keeps the same label as the original dataset in Table 5.3. $ML1$ is trained to discriminate attack (label 1) and benign (label 0) classes. Attack samples with original labels, as shown in Table 5.3, are applied to the multi-class classifier $ML2$ for training. Thus, $ML2$ is trained to discriminate multiple attack types. Consequently, the ML model in the second stage ($ML2$) uses attack samples only (with no benign samples), and reduces the complexity of the problem and in turn, saves computing time.



	BOTNET	BRUTE FORCE	DOS	EXPLOIT	MALWARE	MISC	NORMAL	PROBE	WEBATTACK	WAvg	MAvg
Flow	0.9189	0.9985	0.9985	0.5000	0.9091	0.5714	0.9900	0.9918	0.8571	0.9949	0.8595
Flow+Event	0.9217	0.9981	0.9996	0.7933	0.8781	0.4231	0.9917	0.9919	0.7342	0.9956	0.8591
Flow+Mes	0.8579	0.9978	0.9997	0.7200	0.9091	0.7762	0.9922	0.9928	0.7696	0.9962	0.8906
Flow+Mes+Event	0.8546	0.9978	0.9998	0.7600	0.8781	0.7449	0.9926	0.9964	0.7974	0.9964	0.8913

Figure 5.17: Performance comparison of NIDS (flow-based) and hybrid NIDS and HIDS (flow, event and message-based) under NDSec-1. Mes:message

The trained $ML1$ and $ML2$ use the test dataset separately in the evaluation process. It is similar to the training procedure to re-label different types of attacks in the pre-processing stage with the same label. $ML1$ predicts the binary dataset, filters the samples that are predicted as benign (label 0), and sends the remaining samples estimated as an attack to $ML2$ for detection. Performance evaluation of the proposed method is not straightforward since not all samples are predicted by the multi-class classifiers ($ML2$). Overall performance should consider prediction results for all samples in the test dataset, including the predicted Benign samples by $ML1$ and attacks classified by $ML2$. The left part of Fig. 5.19 illustrates the overall performance measurement process among the entire test procedure. It relies on the outputs of $ML1$ and $ML2$ by combining the binary classification and multi-class classification results. If $ML1$ predicts samples as an attack, these samples are sent to $ML2$ to make an estimation. $ML2$ finally classifies an instance under one of the attack categories. Indeed, if $ML1$ estimates an attack sample as benign, these samples are eliminated prior to $ML2$. At the output of $ML1$, there is no information about the true label of the attack samples that are predicted as Benign by $ML1$. Therefore, it is difficult to obtain the overall performance considering these attack samples. However, by comparing the number of samples predicted by $ML2$ and an original number of samples in test dataset, it is possible to calculate the difference and obtain the true labels. For instance, if there are $60,767$ Bot attacks (label 1) in original test dataset while assuming $60,700$ samples are left for $ML2$ testing, it is apparent that 67 Bot samples are predicted

Table 5.24: Multi-classifier (ML2) performance

	Precision	Recall	F1	Support
Benign	0.0000	0.0000	0.0000	3
Bot	1.0000	1.0000	1.0000	29692
DDOS-HOIC	1.0000	1.0000	1.0000	67449
DDOS-LOIC-HTTP	1.0000	0.9998	0.9999	19166
DDOS-LOIC-UDP	0.9884	1.0000	0.9942	342
DoS-GoldenEye	0.9991	1.0000	0.9996	1163
DoS-Hulk	1.0000	1.0000	1.0000	6553
DoS-SlowHTTPTest	1.0000	1.0000	1.0000	5351
DoS-Slowloris	0.9986	1.0000	0.9993	702
FTP-BruteForce	0.9999	1.0000	1.0000	15918
SSH-Bruteforce	1.0000	1.0000	1.0000	5418
Weighted Avg	0.9999	1.0000	0.9999	151757
Macro Avg	0.9078	0.9091	0.9084	151757

as benign and filtered out by $ML1$. According to this approach, individual prediction performance (e.g., F1 score) can be obtained for all classes under the proposed two-stage cascaded approach.

The CICIDS 2018 dataset with 10 types of attack and 14 types of attack is used to evaluate the cascade system performance separately. In order to compare with the previous work in Section 5.4.1, all 132 flow features, flattened event features (224), and the same matrix of message features ($15 * 50$) are used to evaluate the proposed two-stage cascade machine learning approach. XGBoost is chosen as $ML1$ and $ML2$ in the cascade framework Fig. 5.19. It is worth to note that the base classifier can be replaced with any other machine learning model. The objective of this study is not to propose a new machine learning algorithm but a new framework that can use any machine learning algorithm to boost the detection accuracy of multiple intrusive patterns.

Starting with 10 types of attack CICIDS 2018 dataset, when the test dataset is re-labeled for attack samples using one label instead of the original labels, performance of XGBoost ($ML1$) can be seen in the confusion matrix illustrated in Fig. 5.18. In the confusion matrix in Fig. 5.18, the most benign samples (152169/152172) are predicted correctly and are be filtered by XGBoost (ML1). Specifically, only 3 benign samples are predicted as attacks and sent to $ML2$ for classification. However, there is one (1) attack sample predicted as benign, which is removed by $ML1$ prior to $ML2$.

	Predicted Benign	Predicted Attack
True Benign	152169	3
True Attack	1	151754

Figure 5.18: XGBoost (binary classifier) prediction confusion matrix

Table 5.24 demonstrates classification results of the second stage machine learning model ($ML2$) of XGBoost. By comparing the number of samples in the original test dataset and the classification outputs by $ML2$, the difference for the Benign and Bots categories is $152,169$ and 1, comparing sample support as shown in Table 5.3 and Table 5.24, respectively. Therefore, it is apparent that the eliminated attack sample by $ML1$ belongs to Bot. Table 5.25 illustrates a sample label and prediction results of the two-stage cascaded approach, integrating $ML1$ and $ML2$ prediction results. A total of $151,757$ ($151,754$ and 3) samples as shown in Fig. 5.18 predicted as attack by $ML1$ are sent to $ML2$ for further prediction, which matches the number of samples predicted by $ML2$ as shown in Table 5.25 (sample index from 1 to 151757). These $151,757$ samples final prediction follows $ML2$ estimation results.

Table 5.25 demonstrates that $ML1$ is responsible for part samples (from index $151,758$ to index $303,927$) prediction and $ML2$ makes estimation for the rest of the samples (from index 1 to index $151,757$). Meanwhile, $ML1$ predicts one attack sample as benign and estimates all other benign samples correctly, illustrated in the confusion matrix Fig. 5.18. It is straightforward to calculate overall performance with each sample prediction result and label information.

Fig. 5.20a demonstrates the performance comparison of the proposed cascade system and XGBoost for individual classes and the overall performance (macro average F1 score). The proposed approach increases the macro average F1 score slightly from 0.9993 to 0.9994 and single class DoS-Slowloris from 0.9986 to 0.9993. Since XGBoost achieves remarkably high detection performance for all classes combing flow and host features, the proposed cascaded system does not show critical improvement under CICIDS 2018 dataset with 10 types of attack when rare attack instances are eliminated. However, the benefit of the proposed approach becomes evident when attack types of less/rare instances are included (all 14 types of attacks) as discussed next.

An additional test by including all classes is also conducted adopting the two-stage cascaded machine learning approach. As stated earlier in Section 5.1, CICIDS 2018 contains 14 types of attack. In the previous test, 4 of these attacks (Brute Force-Web, Brute Force-

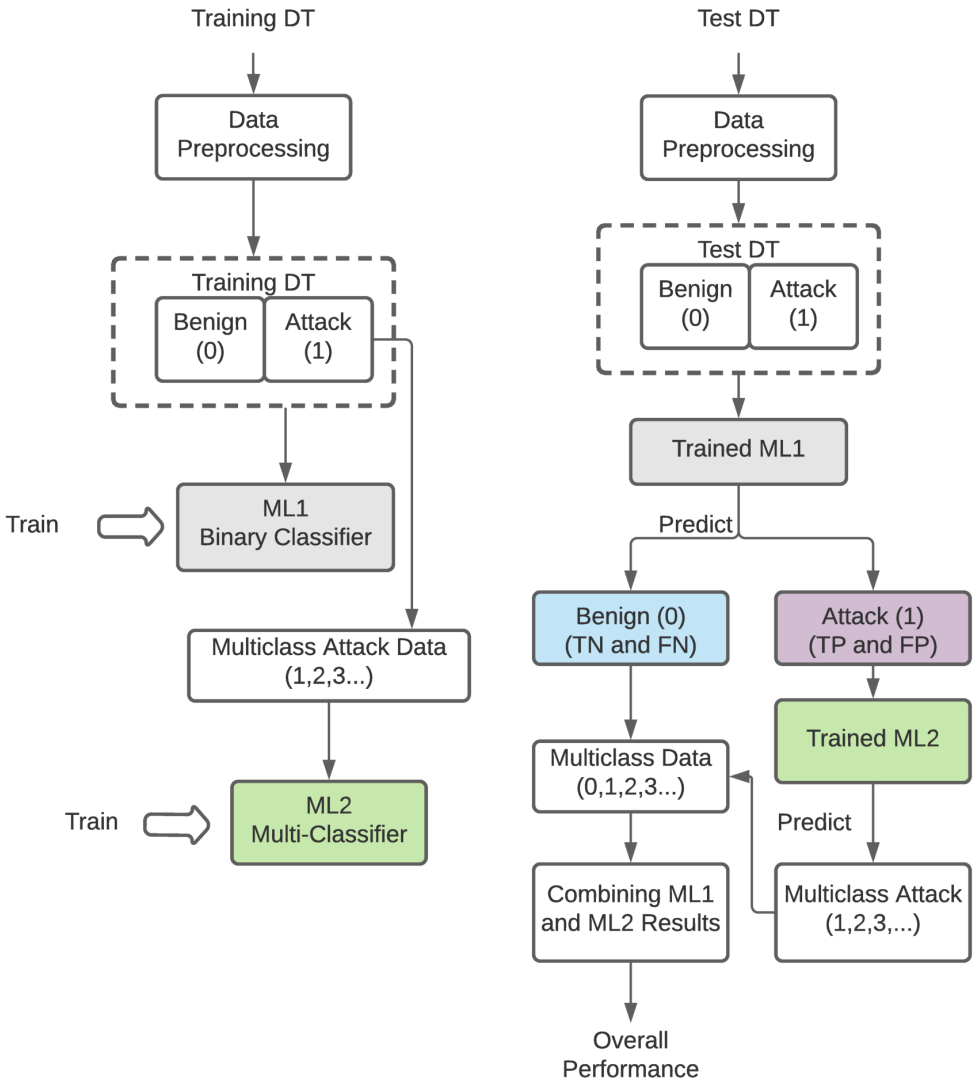
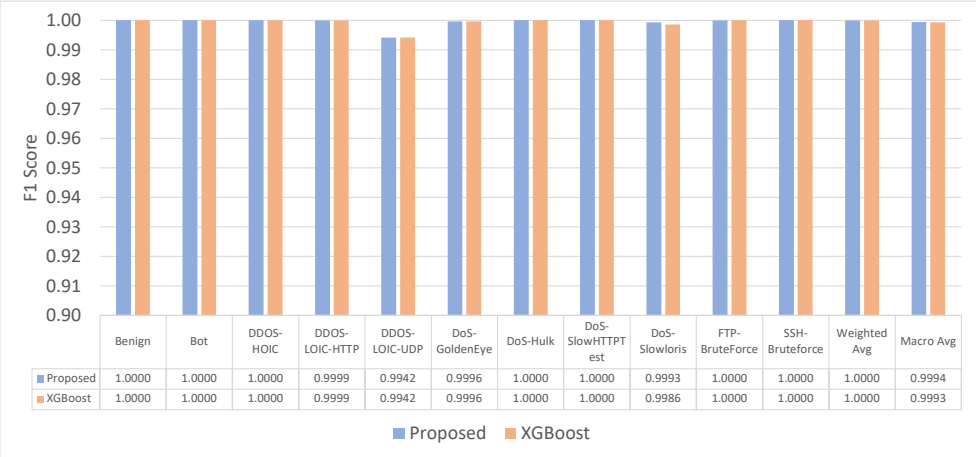
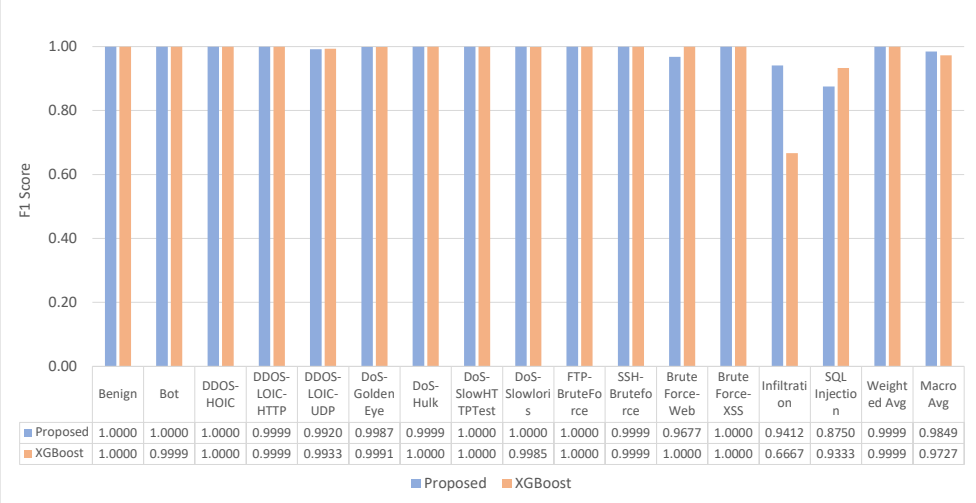


Figure 5.19: Proposed two-stage cascade machine learning model architecture



	Benign	Bot	DDOS-HOIC	DDOS-LOIC-HTTP	DDOS-LOIC-UDP	DoS-GoldenEye	DoS-Hulk	DoS-SlowHTTPTest	DoS-Slowloris	FTP-BruteForce	SSH-Bruteforce	Weighted Avg	Macro Avg
Proposed	1.0000	1.0000	1.0000	0.9999	0.9942	0.9996	1.0000	1.0000	0.9993	1.0000	1.0000	1.0000	0.9994
XGBoost	1.0000	1.0000	1.0000	0.9999	0.9942	0.9996	1.0000	1.0000	0.9986	1.0000	1.0000	1.0000	0.9993

(a) With 10 types of attack



	Benign	Bot	DDOS-HOIC	DDOS-LOIC-HTTP	DDOS-LOIC-UDP	DoS-Golden Eye	DoS-Hulk	DoS-SlowHTTPTest	DoS-Slowloris	FTP-BruteForce	SSH-Bruteforce	Brute Force-Web	Brute Force-XSS	Infiltration	SQL Injection	Weighted Avg	Macro Avg
Proposed	1.0000	1.0000	1.0000	0.9999	0.9920	0.9987	0.9999	1.0000	1.0000	1.0000	0.9999	0.9677	1.0000	0.9412	0.8750	0.9999	0.9849
XGBoost	1.0000	0.9999	1.0000	0.9999	0.9933	0.9991	1.0000	1.0000	0.9985	1.0000	0.9999	1.0000	1.0000	0.6667	0.9333	0.9999	0.9727

(b) With 14 types of attack

Figure 5.20: Performance comparison of the base classifier (XGBoost) and the proposed two-stage cascaded ML framework, under CICIDS 2018 dataset with 10 or 14 types of attack

Table 5.25: All samples in test dataset prediction result and label value integrating ML1 and ML2 estimation results. Attack sample is labeled as 1 and benign is labeled as 0.

	Sample Index	Pred	Label
ML2	1 2 ... 151757	The same as ML2	
ML1	151758	0	1
	151759	0	0
	151760	0	0
	...	0	0
	303927	0	0

XSS, Infiltration, SQL Injection) were eliminated due to being represented by insufficient samples in the dataset (less than 50). For instance, out of these, Infiltration yields an F1 score of 0.6667 under XGBoost due to insufficient samples as shown in Fig. 5.20b. Fig. 5.20b presents test performance using the proposed cascade approach and XGBoost using dataset with insufficient attack samples. From the results, it can be observed that The macro average F1 score (overall performance) increases from 0.9727 to 0.9849. Meanwhile, the proposed approach boosts attack Infiltration performance dramatically, from 0.6667 to 0.9412. The CICIDS 20018 dataset generates Infiltration attack samples via internal infiltration of the network. Specifically, a malware through email is transmitted to the target and exploit a software vulnerability to infiltrate the network from the inside. After successful attack, a backdoor is installed on the victim node, which will enable exploration of the internal system for other susceptible devices. Infiltration attack is critical for the entire network which finds victim nodes from internal network. The proposed cascaded system shows advantages in boosting Infiltration detection performance. However, Brute Force-Web performance decreases by 3% when compared to XGBoost. Considering the benefits of the overall performance and dramatic improvement for the Infiltration attack, the cascade machine learning method for intrusion detection is worthy.

5.4.3 Result analysis

The presented method, utilizing network and host features, demonstrates a more promising performance, in terms of macro average F1 score, that is increased from 0.9246 to 0.9993 under CICIDS 2018. Furthermore, detection performance for all individual attack types has

been shown to improve such as the F1 score of DDOS-LOIC-UDP that was elevated from 0.7609 to 0.9942 and DoS-SlowHTTPTest that improved from 0.5425 to approximately 1 under the CICIDS 2018 dataset. Under the NDSec-1 dataset, with host features, the overall detection performance, in terms of macro average F1 score, has been shown to increase from 0.8595 (when only flow features are used) to 0.8913. The two stage cascaded machine learning framework also improves the system performance slightly whereas it outperforms the base classifier for the individual detection performance (F1 score) of rare classes such as Infiltration whose F1 score has been shown to increase from 0.6667 to 0.9412.

5.5 Conclusion

ML algorithms are extensively applied for network intrusion detection and also in IoT settings. However, the lack of a one-size-fits-all solution calls for novel frameworks for multiple class classification schemes. To this end, a NIDS, namely APWD framework, is proposed that consolidates several ML models, taking advantage of the expertise of certain models for specific attack types to boost detection performance in an IoT network. Experiments have been conducted under a public dataset (NSL-KDD) based on three base detectors (i.e., RF, XGBoost, and Adaboost). Numerical results demonstrate the proposed APWD framework significantly increases the overall accuracy compared to all base detectors. While the overall performance has been boosted, more importantly, the APWD framework has been shown to improve the detection performance for most of the attack types towards the IoT network. The APWD model obtains remarkably high improvement in terms of per-attack type accuracy under the NSL-KDD dataset.

Lately, an NIDS, particularly POLC, extending from APWD, is introduced in this work, that takes advantages of the expertise/competency of particular ML algorithms for certain classes to improve attack detection performance in a network (APWD) and to reduce the decision costs (POLC). Meanwhile, a GAN is deployed to cope with the rare attack samples by augmenting their instances in a systemic manner. Experiments have been conducted to evaluate the presented system performance under NSL-KDD and compared to two individual classifiers (i.e., Decision Tree (DT), XGBoost). It has been shown that the APWD / POLC frameworks determine the same expert ML classifier for each class in NSL-KDD, so they follow the same aggregation steps. Moreover, the proposed models enhance detection performance for most attack groups. For instance, the APWD / POLC model improves the $R2L$ F1 score by up to 192%.

Lastly, NIDS methods have been widely investigated to ensure security via tracking the entire network traffic. However, limitations of NIDS in the recognition of particular

attack types such as Advanced Persistent Threats requires bridging with host-based network intrusion detection systems (HIDS). Basically, HIDS builds on the analysis of various resources in hosts such as logs, files, and folders, without monitoring the network traffic. With this motivation in mind, in this work, a method is proposed, combining HIDS and NIDS, to detect various types of intrusive patterns in a networked environment through a two-stage cascading machine learning (ML) framework. NIDS and HIDS are combined by a feature flattening approach, and a thorough study of the impact of the dimension reduction on the message features at the hosts has been presented. Feature flattening is followed by the introduction of a two-stage cascaded machine learning approach that comprises a binary classifier to discriminate only benign and attack traffic; and a multi-class classifier that discriminates multiple attack types. Under public datasets, CICIDS 2018 and NDSec-1 that contain network and host features (e.g., host event features and host message features), the proposed two-stage intrusion detection framework integrated with feature flattening is evaluated. The numerical results demonstrated that the hybrid network and host features improves the attack detection performance significantly.

Chapter 6

conclusion, Future Directions and Open Issues

6.1 Conclusion

Internet of Things (IoT) has attracted people's attention which is widely used in various applications in recent days. Mobile Crowdsensing (MCS) systems are an internal part of IoT that encounters similar security issues. MCS systems rely on individual users to submit sensing data via smart devices (e.g., smart phones, wearable equipment) with various sensors to collect environment information. This thesis focus on ML-based approaches to detect attack samples in MCS systems. Specifically, this thesis aims to distinguish fake tasks in MCS systems that objects to drain more energy from users' device and clog MCS platform in MCS activities. Fake tasks are harmful for individual participants and MCS platform that deter the development and deployment of MCS technique in real applications. However, in physical systems, it is demanding to address challenges when deploys ML models, such as imbalanced dataset with scarce of intrusion samples and without enough features in original dataset. It is difficult to achieve a promising performance with these issue. With the motivation to tackle imbalanced dataset challenge and obtain an encouraging result, a Deep Belief Network (DBN)-based approach is investigated, composing several oversampling techniques (e.g., Random Over-sampling, Borderline SMOTE and Adaptive Synthetic over-sampling (ADASYN)). The deployed over-sampling methods enlarge synthetic samples using different insertion mechanisms, that result in the generated samples with different properties and features. This leads to performance varying under different oversampling approaches. From numerical results, it is observe that DNB contributes

better detection performance for fake tasks integrating Borderline SMOTE oversampling technique. The second challenge in original MCS dataset generally does not contain enough features, so it is demanding to extract more features for ML models to learning and make good prediction. With this in mind and analyzing MCS dataset characteristics, an unsupervised learning algorithm, namely k-means, is deployed to separate tasks in dataset into several groups. The extracted feature is group information that is labeled with an unique number for every group. It means a task in a group will add a feature with name 'group ID' and fill the feature with value the same as the unique number of this group. Evaluation results demonstrate that a better performance can be achieved than with initial dataset. It means the extracted feature is meaningful and useful for ML models to learn and make wise prediction. The concept can be applied to other research scenarios to build k-means-based approaches extracting more features. Meanwhile, with motivation to improve detection performance and considering prediction costs, an innovated ML framework is introduced that integrates horizontal Federated Learning (FL) to identify fake tasks. The proposed method makes decision with objective to minimize prediction loss.

Furthermore, the submitted tasks are collected and managed conventionally by a centralized MCS platform. A centralized MCS platform is not safe enough to protect and prevent tampering sensing tasks since it confronts the single point of failure which reduces the effectiveness and robustness of MCS system. Meanwhile, fake task attack is a serious threat as it would drain excessive resources from the participant devices and clog the MCS servers to disrupt the services offered by the MCS. In order to address the centralized issue and identify fake tasks, a blockchain-based decentralized MCS is designed. Integration of blockchain into MCS enables a decentralized framework. Moreover, the distributed nature of a blockchain chain prevents sensing tasks from being tampered. The blockchain uses a Practical Byzantine Fault Tolerance (PBFT) consensus that can tolerate 1/3 faulty nodes, making the implemented MCS system robust and sturdy. In addition, an ensemble learning approach is deployed in the blockchain for eliminating fake tasks by malicious requesters. The proposed blockchain-integrated MCS framework endures individual node fault probability to achieve a high Rate of Legitimate tasks ($RoLT$) and a low Rate of Fake tasks ($RoFT$) when compared with a centralized system and Non-Fault Tolerance (NFT) system.

Cyber attackers could compromise the connected devices in IoT network and inject intrusion samples. IoT networks confront security issue due to huge number of connected equipment forming a large attack surface. Machine learning (ML)-based network intrusion detection system (NIDS) is one of most useful techniques to identify network intrusions and ensure IoT system security. An innovative ensemble learning approach, i.e., All Predict Wisest Decides (APWD), is introduced firstly to make a wise decision and achieve a better

attack detection performance. APWD is a general framework that can consist of various ML models or neural networks. It is easy to apply APWD in other applications rather than network intrusion detection. Furthermore, APWD makes a more prudent decision according to individual ML model expertise in specific classes. Numerical results demonstrate APWD outweighs classic ML algorithms in overall performance and single class. Moreover, Predictor Of the Lowest Cost (POLC) is introduced, that extends APWD, making wise decision based on lower cost. Specifically, Thus, both the F1 score (in APWD) and cost (in POLC) determination criteria show advantages in finding suitable expertise for the detection performance in the system. Meanwhile, NIDS faces restrictions in observing the entire traffic information due to encrypted traffic or lack of authority. A host-based intrusion detection system (HIDS) evaluates resources in the host including logs, files, and folders to identify APT attacks that routinely inject malicious files to victimized nodes. The ML-based framework that leverages network and host-based features by running two levels of machine learning algorithms on network and host data. First, a binary classifier is used for detecting benign samples, then all detected attack types undergo a multi-class classifier so to reduce the complexity of the original problem and to improve the overall detection performance. Either APWD / POLC or hybrid HIDS and NIDS system, network intrusion detection accuracy raises that ensures systems security.

6.2 Future Directions and Open Issues

This thesis aims to protect IoT systems in terms of MCS-based applications, blockchain-integrated MCS systems, and IoT networks. Machine learning and deep learning algorithms are utilized and investigated intensively to ensure system security and privacy. There are still ongoing works in the discussed topics (e.g., fake task detection in MCS, blockchain-integrated decentralized MCS framework, network intrusion detection in IoT). I will continue these research topics if I have any chances and opportunities in the future.

Firstly, in order to improve fake task detection performance, several ML models and one deep learning model (DBN) are investigated. Considering deep learning algorithms with powerful learning skills with high-level features from data in an incremental manner, in the future, it is valuable to implement detection system based on other deep network architectures (e.g., CNN, Deep Neural Network (DNN)) in order to further improve the accuracy and precision performance of the detection of the fake sensing tasks submitted to MCS platforms. In [237], the authors enhance both detecting rate and accuracy of malware by relying on a CNN algorithm. The study in [251] demonstrates multiple deep learning methods (e.g., autoencoder, DNN, CNN, and deep reinforcement learning) to

detect attacks. In particular, to boost the prediction accuracy of and reduce detection time, a deep Q network-based attack detection method is designed integrating CNN compressed the state space to reduce training time. Higher detection performance also leads to lower average loss in terms of task values for dynamic reputation and vote based federated models. Meanwhile, the challenge with MCS is not only participation of untrusted parties but also the vulnerabilities of the distributed sensing system (i.e., fake sensed data) as introduced in [199]. It is worth investigating data poisoning attacks in the future.

Secondly, this thesis investigates saving legitimate tasks and the elimination of fake tasks in MCS systems via blockchain networks and ensemble learning, respectively. Within the designed architecture, incentive mechanism [254] is not considered to promote individual block node participation in PBFT consensus. The proposed framework does not involve reward distribution in the blockchain under the assumption of all nodes being 100% willing to vote (except the faulty nodes) in PBFT consensus. The rewards are normally used to incentivize nodes to contribute to the voting process during PBFT. Thus, rewards may result in different voting rates in PBFT. Currently, the voting rate in the proposed framework only relates to the number of active nodes and the total number of nodes. When rewards are considered for individual nodes, the design architecture will need to be revisited and updates. Therefore, under the current framework, it is difficult to evaluate the blockchain performance in terms of rewards distribution and transaction fees. Future research agenda includes updating the current framework with a new blockchain-enables MCS design by taking the other metrics into account, such as rewards and transaction fees, and throughput. Furthermore, hash rate on the blockchain is calculated using the SHA-256 algorithm. In the extension of this study, we will deploy different algorithms (e.g., SH384 or SHA512 [123]), that may impact blockchain network performance such as average block time and hash rate. Meanwhile, it is worth noting that there exists a trade-off between consensus time and performance with respect to the size of the blockchain network. Thus, the system performance improves with the increasing number of nodes in the blockchain whereas the PBFT consensus protocol requires longer duration to achieve consensus for larger number of blocks in the blockchain. In light of this observation, our ongoing work involves addressing the time cost for PBFT consensus constrained to the scale of the blockchain network.

Thirdly, ongoing works for network intrusion detection in IoT systems, include integrating CNN into APWD / POLC as base learners since APWD / POLC demonstrates a lower performance than CNN model [62]. APWD / POLC is a generic framework that does not limit the number of base estimators and the kinds of base estimators. Currently, only fewer base estimators are included so our future work will increase the number of base learners and raise diversity of the type of base learners (e.g., deep neural networks,

unsupervised algorithms, and non tree-based supervised algorithms) in APWD / POLC framework. Meanwhile, billions of electronic devices (e.g., sensors) and machines are being connected to IoT systems. This connectivity trend is anticipated to continue well into the future, especially regarding wireless IoT in applications such as smart cities. It means each piece of equipment added to the network has multiple chances to be a vector for Zero-day attacks. Therefore, ML algorithms should have the ability to learn adaptive and continuously to alleviate these anticipated attacks given the scale of network entry points. The current ML paradigm works in two steps: first, a specific training dataset runs a machine-learning algorithm to construct a model, and then the model is used for its intended IoT application [52]. The process of continuous ML is still in early development, so challenges exist in applying continuous learning to IoT. In [215], the authors demonstrate lifelong ML taking into account systems that can learn various tasks from one or more domains during their lifetime. The objective is to absorb learned sequentially to selectively apply that knowledge when learning a new scheme to form more refined assumptions and/or policies. Moreover, the authors suggest that the AI community should look beyond merely one-time learning algorithms to more seriously consider the nature of systems that can learn throughout their operational lifetime. Consequently, in the future, we can consider ways and means to deploy lifelong ML to identify cyber intrusion in the future in IoT.

References

[1] A survey on cybersecurity, data privacy, and policy issues in cyber-physical system deployments in smart cities. *Sustainable Cities and Society*, 50:101660, 2019.

[2] G. Dahl B. Ramabhadran G. E. Hinton A. Mohamed, T. N. Sainath and M. A. Picheny. Deep belief networks using discriminative features for phone recognition. *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5060–5063, 2011.

[3] G. Hinton A. Mohamed and G. Penn. Understanding how deep belief networks perform acoustic modelling. *2012 IEEE International Conf. on Acoustics,Speech and Signal Processing*, pages 4273–4276, 2012.

[4] Zeeshan Abbas and Wonyong Yoon. A survey on energy conserving mechanisms for the internet of things: Wireless networking aspects. *Sensors*, 15(10):24818–24847, 2015.

[5] Saveen A Abeyratne and Radmehr Monfared. Blockchain ready manufacturing supply chain using distributed ledger. 9 2016.

[6] Nyoman Abiwinanda, Muhammad Hanif, S Tafwida Hesaputra, Astri Handayani, and Tati Rajab Mengko. Brain tumor classification using convolutional neural network. In *World congress on medical physics and biomedical engineering 2018*, pages 183–189. Springer, 2019.

[7] Qasem Abu Al-Haija and Abdelraouf Ishtaiwi. Multiclass classification of firewall log files using shallow neural network for network security applications. In *Soft Computing for Security Applications*, pages 27–41. Springer, 2022.

[8] Fadele Ayotunde Alaba, Mazliza Othman, Ibrahim Abaker Targio Hashem, and Faiz Alotaibi. Internet of things security: A survey. *Journal of Network and Computer Applications*, 88:10–28, 2017.

[9] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6. Ieee, 2017.

[10] Ikram Ali, Mwitende Gervais, Emmanuel Ahene, and Fagen Li. A blockchain-based certificateless public key signature scheme for vehicle-to-infrastructure communication in vanets. *Journal of Systems Architecture*, 99:101636, 2019.

[11] Muhammad Salek Ali, Koustabh Dolui, and Fabio Antonelli. Iot data privacy via blockchains and ipfs. In *Proceedings of the Seventh International Conference on the Internet of Things*, IoT '17, New York, NY, USA, 2017. Association for Computing Machinery.

[12] S. Ali, G. Wang, M. Z. A. Bhuiyan, and H. Jiang. Secure data provenance in cloud-centric internet of things via blockchain smart contracts. In *IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, pages 991–998, 2018.

[13] Osama Alkadi, Nour Moustafa, Benjamin Turnbull, and Kim-Kwang Raymond Choo. A deep blockchain framework-enabled collaborative intrusion detection for protecting iot and cloud networks. *IEEE Internet of Things Journal*, 8(12):9463–9472, June 2020.

[14] Hamed Alqahtani, Manolya Kavakli-Thorne, and Gulshan Kumar. Applications of generative adversarial networks (gans): An updated review. *Archives of Computational Methods in Engineering*, 28(2):525–552, 2021.

[15] Esra'a Alshdaifat, Malak Al-hassan, and Ahmad Aloqaily. Effective heterogeneous ensemble classification: An alternative approach for selecting base classifiers. *ICT Express*, 2020.

[16] Amar Amouri, Vishwa T Alaparthy, and Salvatore D Morgera. A machine learning based intrusion detection system for mobile internet of things. *Sensors*, 20(2):461, 2020.

[17] Farzana Anowar, Samira Sadaoui, and Bassant Selim. Conceptual and empirical comparison of dimensionality reduction algorithms (pca, kpca, lda, mds, svd, lle, isomap, le, ica, t-sne). *Computer Science Review*, 40:100378, 2021.

[18] Mohamad Arafeh, May El Barachi, Azzam Mourad, and Fatna Belqasmi. A blockchain based architecture for the detection of fake sensing in mobile crowdsensing. In *2019 4th International Conference on Smart and Sustainable Technologies (SpliTech)*, pages 1–6. IEEE, 2019.

[19] Kevin Ashton et al. That 'internet of things' thing. *RFID journal*, 22(7):97–114, 2009.

[20] Ahmed Ben Ayed. A conceptual secure blockchain-based electronic voting system. *International Journal of Network Security & Its Applications*, 9(3):01–09, 2017.

[21] Miloud Bagaa, Tarik Taleb, Jorge Bernal Bernabe, and Antonio Skarmeta. A machine learning security framework for iot systems. *IEEE Access*, 8:114066–114077, 2020.

[22] S. Bajoudah, C. Dong, and P. Missier. Toward a decentralized, trust-less marketplace for brokered iot data trading using blockchain. In *IEEE International Conference on Blockchain (Blockchain)*, pages 339–346, 2019.

[23] Thar Baker, Muhammad Asim, Áine MacDermott, Farkhund Iqbal, Faouzi Kamoun, Babar Shah, Omar Alfandi, and Mohammad Hammoudeh. A secure fog-based platform for scada-based iot critical infrastructure. *Software: Practice and Experience*, 50(5):503–518, 2020.

[24] J. Ballesteros, B. Carbunar, M. Rahman, N. Rishe, and S. S. Iyengar. Towards safe cities: A mobile and social networking approach. *IEEE Trans. on Parallel and Dist. Sys.*, 25/9:2451–2462, Sep. 2014.

[25] Marco Bazzani, Davide Conzon, Andrea Scalera, Maurizio A Spirito, and Claudia Irene Trainito. Enabling the iot paradigm in e-health solutions through the virtus middleware. In *2012 IEEE 11th international conference on trust, security and privacy in computing and communications*, pages 1954–1959. IEEE, 2012.

[26] Frank Beer, Tim Hofer, David Karimi, and Ulrich Bühler. A new attack composition for network security. In Paul Müller, Bernhard Neumair, Helmut Raiser, and Gabi Dreo Rodosek, editors, *10. DFN-Forum Kommunikationstechnologien*, pages 11–20, Bonn, 2017. Gesellschaft für Informatik e.V.

[27] Iddo Bentov, Ariel Gabizon, and Alex Mizrahi. Cryptocurrencies without proof of work. In *International conference on financial cryptography and data security*, pages 142–157. Springer, 2016.

[28] Elham Besharati, Marjan Naderan, and Ehsan Namjoo. Lr-hids: logistic regression host-based intrusion detection system for cloud environments. *Journal of Ambient Intelligence and Humanized Computing*, 10(9):3669–3692, 2019.

[29] Thomas Bocek and Burkhard Stiller. Smart contracts–blockchains in the wings. In *Digital Marketplaces Unleashed*, pages 169–184. Springer, 2018.

[30] Raveendranadh Bokka and Tamilselvan Sadasivam. Deep learning model for detection of attacks in the internet of things based smart home environment. In *Proceedings of International Conference on Recent Trends in Machine Learning, IoT, Smart Cities and Applications*, pages 725–735. Springer, 2021.

[31] Djallel Eddine Boubiche, Muhammad Imran, Aneela Maqsood, and Muhammad Shoaib. Mobile crowd sensing–taxonomy, applications, challenges, and solutions. *Computers in Human Behavior*, 101:352–370, 2019.

[32] Sarah Bouraga. A taxonomy of blockchain consensus protocols: A survey and classification framework. *Expert Systems with Applications*, 168:114384, 2021.

[33] Ioannis Boutsis and Vana Kalogeraki. Privacy preservation for participatory sensing data. In *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 103–113, 2013.

[34] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[35] Matteo Cagnazzo, Markus Hertlein, Thorsten Holz, and Norbert Pohlmann. Threat modeling for mobile health systems. In *2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pages 314–319. IEEE, 2018.

[36] C. Cai, Y. Zheng, Y. Du, Z. Qin, and C. Wang. Towards private, robust, and verifiable crowdsensing systems via public blockchains. *IEEE Transactions on Dependable and Secure Computing*, pages 1–1, 2019.

[37] A. Capponi, C. Fiandrino, B. Kantarci, L. Foschini, D. Kliazovich, and P. Bouvry. A survey on mobile crowdsensing systems: Challenges, solutions, and opportunities. *IEEE Communications Surveys Tutorials*, 21(3):2419–2465, thirdquarter 2019.

[38] A. Capponi, C. Fiandrino, D. Kliazovich, P. Bouvry, and S. Giordano. A cost-effective distributed framework for data collection in cloud-based mobile crowd sensing architectures. *IEEE Transactions on Sustainable Computing*, 2(1):3–16, Jan 2017.

[39] Andrea Capponi, Claudio Fiandrino, Burak Kantarci, Luca Foschini, Dzmitry Kliazovich, and Pascal Bouvry. A survey on mobile crowdsensing systems: Challenges, solutions, and opportunities. *IEEE communications surveys & tutorials*, 21(3):2419–2465, 2019.

[40] Lyn Carson, Professorial Fellow Lyn Carson, and Brian Martin. *Random selection in politics*. Greenwood Publishing Group, 1999.

[41] Dimitris Chatzopoulos, Sujit Gujar, Boi Faltings, and Pan Hui. Privacy preserving and cost optimal mobile crowdsensing using smart contracts on blockchain. In *IEEE 15th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pages 442–450, 2018.

[42] Ashima Chawla, Brian Lee, Sheila Fallon, and Paul Jacob. Host based intrusion detection system with combined cnn/rnn model. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 149–158. Springer, 2018.

[43] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[44] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[45] Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, and Yuan Tang. Xgboost: extreme gradient boosting. *R package version 0.4-2*, pages 1–4, 2015.

[46] Xuankai Chen, Murat Simsek, and Burak Kantarci. Locally reconfigurable self organizing feature map for high impact malicious tasks submission in mobile crowdsensing. *Internet of Things*, 12:100297, 2020.

[47] Zhiyan Chen, Claudio Fiandrino, and Burak Kantarci. On blockchain integration into mobile crowdsensing via smart embedded devices: A comprehensive survey. *Journal of Systems Architecture*, page 102011, 2021.

[48] Zhiyan Chen, Jinxin Liu, Yu Shen, Murat Simsek, Burak Kantarci, Hussein T. Mouftah, and Petar Djukic. Machine learning-enabled iot security: Open issues and challenges under advanced persistent threats. *ACM Comput. Surv.*, apr 2022. Just Accepted.

[49] Zhiyan Chen, Murat Simsek, Burak Kantarci, and Petar Djukic. All predict wisest decides: A novel ensemble method to detect intrusive traffic in iot networks. In *2021 IEEE Global Communications Conference (GLOBECOM)*, pages 01–06. IEEE, 2021.

[50] Zhiyan Chen, Yueqian Zhang, Murat Simsek, and Burak Kantarci. Deep belief network-based fake task mitigation for mobile crowdsensing under data scarcity. In *IEEE Intl. Conf. on Communications (ICC)*, pages 1–7, 2020.

[51] Zhiyan Chen, Yueqian Zhang, Murat Simsek, and Burak Kantarci. Deep belief network-based fake task mitigation for mobile crowdsensing under data scarcity. *IEEE Intl. Conf. on Communications*, June 2020 (accepted).

[52] Zhiyuan Chen and Bing Liu. Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–207, 2018.

[53] Mauro Conti, E Sandeep Kumar, Chhagan Lal, and Sushmita Ruj. A survey on security and privacy issues of bitcoin. *IEEE Communications Surveys & Tutorials*, 20(4):3416–3452, 2018.

[54] Li Da Xu. Enterprise systems: state-of-the-art and future trends. *IEEE Transactions on Industrial Informatics*, 7(4):630–640, 2011.

[55] Li Da Xu, Wu He, and Shancang Li. Internet of things in industries: A survey. *IEEE Transactions on industrial informatics*, 10(4):2233–2243, 2014.

[56] Abbas Abou Daya, Mohammad A. Salahuddin, Noura Limam, and Raouf Boutaba. BotChase: Graph-based bot detection using machine learning. 17(1):15–29. Conference Name: IEEE Transactions on Network and Service Management.

[57] Xavier Decuyper. How does a blockchain work - simply expained. 11 2017.

[58] Hans-Peter Deutsch and Mark W Beinker. Principal component analysis. In *Derivatives and Internal Models*, pages 793–804. Springer, 2019.

[59] Preethi Devan and Neelu Khare. An efficient xgboost–dnn-based classification model for network intrusion detection system. *Neural Computing and Applications*, 32(16):12499–12514, 2020.

[60] L Dhanabal and SP Shantharajah. A study on NSL-KDD dataset for intrusion detection system based on classification algorithms. *Int. J. of Advanced Research in Computer and Communication Engineering*, 4/6:446–452, 2015.

[61] Thomas G Dietterich et al. Ensemble learning. *The handbook of brain theory and neural networks*, 2:110–125, 2002.

[62] Yalei Ding and Yuqing Zhai. Intrusion detection system for nsl-kdd dataset using convolutional neural networks. In *Intl Conference on Computer Science and Artificial Intelligence*, pages 81–85, 2018.

[63] H. Duan, Y. Zheng, Y. Du, A. Zhou, C. Wang, and M. H. Au. Aggregating crowd wisdom via blockchain: A private, correct, and robust realization. In *IEEE International Conference on Pervasive Computing and Communications (PerCom*, pages 1–10, 2019.

[64] Shannon Eggers. A novel approach for analyzing the nuclear supply chain cyber-attack surface. *Nuclear Engineering and Technology*, 53(3):879–887, 2021.

[65] Reda M Elbasiony, Elsayed A Sallam, Tarek E Eltobely, and Mahmoud M Fahmy. A hybrid network intrusion detection framework based on random forests and weighted k-means. *Ain Shams Engineering Journal*, 4(4):753–762, 2013.

[66] Faris Elghaish, M Reza Hosseini, Sandra Matarneh, Saeed Talebi, Song Wu, Igor Martek, Mani Poshdar, and Nariman Ghodrati. Blockchain and the 'internet of things' for the construction industry: research trends and opportunities. *Automation in Construction*, 132:103942, December 2021.

[67] Wisam Elmasry, Akhan Akbulut, and Abdul Halim Zaim. Evolving deep learning architectures for network intrusion detection using a double PSO metaheuristic. *Computer Networks*, 168:107042, February 2020.

[68] Nabeil Eltayieb, Rashad Elhabob, Alzubair Hassan, and Fagen Li. A blockchain-based attribute-based signcryption scheme to secure data sharing in the cloud. *Journal of Systems Architecture*, 102:101653, 2020.

[69] Justin Engelmann and Stefan Lessmann. Conditional wasserstein gan-based oversampling of tabular data for imbalanced learning. *Expert Systems with Applications*, 174:114582, 2021.

[70] Shaohan Feng, Wenbo Wang, Dusit Niyato, Dong In Kim, and Ping Wang. Competitive data trading in wireless-powered internet of things (iot) crowdsensing systems with blockchain. In *IEEE International Conference on Communication Systems (ICCS)*, pages 289–394, 2018.

[71] Zhenni Feng and Junchang Chen. Blockchain based mobile crowd sensing for reliable data sharing in iot systems. In *2021 IFIP Networking Conference (IFIP Networking)*, pages 1–3. IEEE, 2021.

[72] C. Fiandrino, A. Capponi, G. Cacciatore, D. Kliazovich, U. Sorger, P. Bouvry, B. Kantarci, F. Granelli, and S. Giordano. CrowdSenSim: a simulation platform for mobile crowdsensing in realistic urban environments. *IEEE Access*, 5:3490–3503, 2017.

[73] Raghu K Ganti, Fan Ye, and Hui Lei. Mobile crowdsensing: current state and future challenges. *IEEE Communications Magazine*, 49(11), 2011.

[74] Sheng Gao, Xiuhua Chen, Jianming Zhu, Xuewen Dong, and Jianfeng Ma. Trustworker: A trustworthy and privacy-preserving worker selection scheme for blockchain-based crowdsensing. *IEEE Transactions on Services Computing*, pages 1–1, (early access) 2021.

[75] Xianwei Gao, Chun Shan, Changzhen Hu, Zequn Niu, and Zhen Liu. An adaptive ensemble machine learning model for intrusion detection. *IEEE Access*, 7:82512–82521, 2019.

[76] Robin Gassais, Naser Ezzati-Jivan, Jose M Fernandez, Daniel Aloise, and Michel R Dagenais. Multi-level host-based intrusion detection system for internet of things. *Journal of Cloud Computing*, 9(1):1–16, 2020.

[77] Nastassja Gaudet, Abhijeet Sahu, Ana E Goulart, Edmond Rogers, and Kate Davis. Firewall configuration and path analysis for smartgrid networks. In *2020 IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR)*, pages 1–6. IEEE, 2020.

[78] Chunpeng Ge, Xinshu Ma, and Zhe Liu. A semi-autonomous distributed blockchain-based framework for uavs system. *Journal of Systems Architecture*, 107:101728, 2020.

[79] Simon Osindero Geoffrey E. Hinton and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 8/7:1527–1554, 2006.

[80] S. Gisdakis, T. Giannetsos, and P. Papadimitratos. Security, privacy, and incentive provision for mobile crowd sensing systems. *IEEE Internet of Things Journal*, 3(5):839–853, Oct. 2016.

[81] Vincent Gramoli. From blockchain consensus back to byzantine consensus. *Future Generation Computer Systems*, 107:760–769, 2020.

[82] Y. Guo, B. Guo, Y. Liu, Z. Wang, Y. Ouyang, and Z. Yu. Crowdsafe: Detecting extreme driving behaviors based on mobile crowdsensing. In *IEEE SmartWorld*, pages 1–8, Aug 2017.

[83] Edwardo A. Garcia Shutao Li Haibo He, Yang Bai. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, page 2008, 2008.

[84] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-smote: A new over-sampling method in imbalanced data sets learning. pages 878–887, 2005.

[85] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *International conference on intelligent computing*, pages 878–887. Springer, 2005.

[86] R. Han, N. Foutris, and C. Kotselidis. Demystifying crypto-mining: Analysis and optimizations of memory-hard pow algorithms. In *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 22–33, 2019.

[87] Ch Gayathri Harshitha, M Kameswara Rao, and P Neelesh Kumar. A novel mechanism for host-based intrusion detection system. In *First International Conference on Sustainable Technologies for Computational Intelligence*, pages 527–536. Springer, 2020.

[88] Mahmudul Hasan, Md. Milon Islam, Md Ishrak Islam Zarif, and M. M. A. Hashem. Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches. *Internet of Things*, 7:100059, September 2019.

[89] Xinhong Hei, Xinyue Yin, Yichuan Wang, Ju Ren, and Lei Zhu. A trusted feature aggregator federated learning for distributed malicious attack detection. *Computers & Security*, 99:102033, December 2020.

[90] GE Hinton, P Dayan, BJ Frey, and RM Neal. The "wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161, 1995.

[91] Geoffrey E Hinton. Deep belief networks. *Scholarpedia*, 4(5):5947, 2009.

[92] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[93] Jiejun Hu, Kun Yang, Kezhi Wang, and Kai Zhang. A blockchain-based reward mechanism for mobile crowdsensing. *IEEE Transactions on Computational Social Systems*, 7(1):178–191, 2020.

[94] Junqin Huang, Linghe Kong, Hong-Ning Dai, Weiping Ding, Long Cheng, Guihai Chen, Xi Jin, and Peng Zeng. Blockchain-based mobile crowd sensing in industrial systems. *IEEE Trans. on Industrial Informatics*, 16(10):6553–6563, Oct 2020.

[95] P. Huang, X. Zhang, L. Guo, and M. Li. Incentivizing crowdsensing-based noise monitoring with differentially-private locations. *IEEE Transactions on Mobile Computing*, pages 1–1, 2019.

[96] Zonghao Huang, Miao Pan, and Yanmin Gong. Robust truth discovery against data poisoning in mobile crowdsensing. In *IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, 2019.

[97] Abiodun M Ikotun, Absalom E Ezugwu, Laith Abualigah, Belal Abuhaija, and Jia Heming. K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data. *Information Sciences*, 2022.

[98] Theodoros Iliou, Christos-Nikolaos Anagnostopoulos, Ioannis M Stephanakis, and George Anastassopoulos. A novel data preprocessing method for boosting neural network performance: a case study in osteoporosis prediction. *Information Sciences*, 380:92–100, 2017.

[99] Abbas Javed, Hadi Larijani, and Andrew Wixted. Improving energy consumption of a commercial building with iot and machine learning. *IT Professional*, 20(5):30–38, 2018.

[100] Bin Jia and Yongquan Liang. Anti-d chain: A lightweight ddos attack detection scheme based on heterogeneous ensemble learning in blockchain. *China Communications*, 17(9):11–24, September 2020.

[101] Kaiyuan Jiang, Wenya Wang, Aili Wang, and Haibin Wu. Network intrusion detection combined hybrid sampling with deep hierarchical network. *IEEE Access*, 8:32464–32476, 2020.

[102] Shuhao Jiang, Jiajun Li, Shijun Gong, Junchao Yan, Guihai Yan, Yi Sun, and Xiaowei Li. Bzip: A compact data memory system for utxo-based blockchains. *Journal of Systems Architecture*, page 101809, 2020.

[103] Iain M Johnstone and Arthur Yu Lu. Sparse principal components analysis. *arXiv preprint arXiv:0901.4392*, 2009.

[104] Maha Kadadha, Hadi Otrok, Rabeb Mizouni, Shakti Singh, and Anis Ouali. Sensechain: A blockchain-based crowdsensing framework for multiple requesters and multiple workers. *Future Generation Computer Systems*, 105:650–664, 2020.

[105] Mehran Kafai and Bir Bhanu. Dynamic bayesian networks for vehicle classification in video. *IEEE Transactions on Industrial Informatics*, 8(1):100–109, 2011.

[106] Jiawen Kang, Zehui Xiong, Dusit Niyato, Yuze Zou, Yang Zhang, and Mohsen Guizani. Reliable federated learning for mobile networks. *IEEE Wireless Communications*, 27(2):72–80, 2020.

[107] V Kanimozhi and T Prem Jacob. Artificial intelligence based network intrusion detection with hyper-parameter optimization tuning on the realistic cyber dataset cse-cic-ids2018 using cloud computing. In *2019 international conference on communication and signal processing (ICCSP)*, pages 0033–0036. IEEE, 2019.

[108] Atreyi Kankanhalli, Yannis Charalabidis, and Sehl Mellouli. Iot and ai for smart government: A research agenda, 2019.

[109] B. Kantarci and H. T. Mouftah. Trustworthy sensing for public safety in cloud-centric internet of things. *IEEE Internet of Things Journal*, 1(4):360–368, Aug. 2014.

[110] Sydney Mambwe Kasongo and Yanxia Sun. A deep learning method with filter based feature engineering for wireless intrusion detection system. 7:38597–38607. Conference Name: IEEE Access.

[111] Jasmin Kevric, Samed Jukic, and Abdulhamit Subasi. An effective combining classifier approach using tree algorithms for network intrusion detection. *Neural Computing and Applications*, 28(1):1051–1058, 2017.

[112] Fazlullah Khan, Ateeq Ur Rehman, Jiangbin Zheng, Mian Ahmad Jan, and Muhammad Alam. Mobile crowdsensing: A survey on privacy-preservation, task management, assignment models, and incentives mechanisms. *Future Generation Computer Systems*, 100:456 – 472, 2019.

[113] Chayoung Kim and JiSu Park. Designing online network intrusion detection using deep auto-encoder Q-learning. *Computers & Electrical Engineering*, 79:106460, October 2019.

[114] Myoung-Jong Kim, Dae-Ki Kang, and Hong Bae Kim. Geometric mean based boosting algorithm with over-sampling to resolve data imbalance problem for bankruptcy prediction. *Expert Systems with Applications*, 42(3):1074–1082, 2015.

[115] Sunghwan Kim, Seunghyun Yoon, Jargalsaikhan Narantuya, and Hyuk Lim. Secure collecting, optimizing, and deploying of firewall rules in software-defined networks. *IEEE Access*, 8:15166–15177, 2020.

[116] Kenneth Kimani, Vitalice Oduol, and Kibet Langat. Cyber security challenges for iot-based smart grid networks. *International Journal of Critical Infrastructure Protection*, 25:36–49, 2019.

[117] Djamel Eddine Kouicem, Abdelmadjid Bouabdallah, and Hicham Lakhlef. Internet of things security: A top-down survey. *Computer Networks*, 141:199–221, 2018.

[118] K Krishna and M Narasimha Murty. Genetic k-means algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 29(3):433–439, 1999.

[119] Ioannis Krontiris and Tassos Dimitriou. Privacy-respecting discovery of data providers in crowd-sensing applications. In *IEEE International Conference on Distributed Computing in Sensor Systems*, pages 249–257, 2013.

[120] N. Kshetri and J. Voas. Blockchain-enabled e-voting. *IEEE Software*, 35(4):95–99, July 2018.

[121] G. Kumar, R. Saha, M. K. Rai, R. Thomas, and T. Kim. Proof-of-work consensus approach in blockchain technology for cloud and fog computing using maximization-factorization statistics. *IEEE Internet of Things Journal*, 6(4):6835–6842, 2019.

[122] Tomasz Kuśmierczyk, Joseph Sakaya, and Arto Klami. Variational bayesian decision-making for continuous utilities. In *Advances in Neural Information Processing Systems*, pages 6395–6405, 2019.

[123] Alexandr Kuznetsov, Kyryl Shekhanin, Andrii Kolhatin, Diana Kovalchuk, Vitalina Babenko, and Iryna Perevozova. Performance of hash algorithms on gpus for use in blockchain. In *2019 IEEE International Conference on Advanced Trends in Information Theory (ATIT)*, pages 166–170. IEEE, 2019.

[124] Badr Lahasn and Hussein Samma. Optimized deep autoencoder model for internet of things intruder detection. *IEEE Access*, 2022.

[125] Nicholas D Lane, Yohan Chon, Lin Zhou, Yongzhe Zhang, Fan Li, Dongwon Kim, Guanzhong Ding, Feng Zhao, and Hojung Cha. Piggyback crowdsensing (pcs): Energy efficient crowdsourcing of mobile sensor data by exploiting smartphone app opportunities. 2013.

[126] Shahid Latif, Zeba Idrees, Jawad Ahmad, Lirong Zheng, and Zhuo Zou. A blockchain-based architecture for secure and trustworthy operations in the industrial internet of things. *Journal of Industrial Information Integration*, 21:100190, March 2021.

[127] Wenke Lee, Wei Fan, Matthew Miller, Salvatore J Stolfo, and Erez Zadok. Toward cost-sensitive modeling for intrusion detection and response. *Journal of computer security*, 10(1-2):5–22, 2002.

[128] Meng Li, Liehuang Zhu, and Xiaodong Lin. Privacy-preserving traffic monitoring with false report filtering via fog-assisted vehicular crowdsensing. *IEEE Transactions on Services Computing*, 2019.

[129] Ming Li, Jian Weng, Anjia Yang, Wei Lu, Yue Zhang, Lin Hou, Jia-Nan Liu, Yang Xiang, and Robert H Deng. Crowdbc: A blockchain-based decentralized framework for crowdsourcing. *IEEE Transactions on Parallel and Distributed Systems*, 30(6):1251–1266, 2018.

[130] Mohan Li, Yanbin Sun, Hui Lu, Sabita Maharjan, and Zhihong Tian. Deep reinforcement learning for partially observable data poisoning attack in crowdsensing systems. *IEEE Internet of Things Journal*, 2019.

[131] R. Li, T. Song, B. Mei, H. Li, X. Cheng, and L. Sun. Blockchain for large-scale internet of things data storage and protection. *IEEE Transactions on Services Computing*, 12(5):762–771, 2019.

[132] Wenyu Li, Chenglin Feng, Lei Zhang, Hao Xu, Bin Cao, and Muhammad Ali Imran. A scalable multi-layer pbft consensus for blockchain. *IEEE Transactions on Parallel and Distributed Systems*, 32(5):1146–1160, 2020.

[133] Xuran Li, Qiu Wang, Hong-Ning Dai, and Hao Wang. A novel friendly jamming scheme in industrial crowdsensing networks against eavesdropping attack. *Sensors*, 18(6):1938, 2018.

[134] Yixin Li, Zhen Wang, Jia Fan, Yue Zheng, Yili Luo, Chunhua Deng, and Jianwei Ding. An extensible consensus algorithm based on pbft. In *2019 International conference on cyber-enabled distributed computing and knowledge discovery (CyberC)*, pages 17–23. IEEE, 2019.

[135] Yuxi Li, Liang Qiao, and Zhihan Lv. An optimized byzantine fault tolerance algorithm for consortium blockchain. *Peer-to-Peer Networking and Applications*, pages 1–14, 2021.

[136] Danwei Liang, Jian An, Jindong Cheng, He Yang, and Ruowei Gui. The quality control in crowdsensing based on twice consensuses of blockchain. In *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*, pages 630–635, 2018.

[137] Junwei Liang and Maode Ma. Co-maintained database based on blockchain for idss: A lifetime learning framework. *IEEE Transactions on Network and Service Management*, 2021.

[138] X. Liang, S. Shetty, D. Tosh, C. Kamhoua, K. Kwiat, and L. Njilla. Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability. In *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pages 468–477.

[139] X. Liang, J. Zhao, S. Shetty, and D. Li. Towards data assurance and resilience in iot using blockchain. In *MILCOM IEEE Military Communications Conference (MILCOM)*, pages 261–266, 2017.

[140] Jie Lin, Wei Yu, Nan Zhang, Xinyu Yang, Hanlin Zhang, and Wei Zhao. A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications. *IEEE Internet of Things Journal*, 4(5):1125–1142, 2017.

[141] Debin Liu and L Jean Camp. Proof of work can work. In *WEIS*, 2006.

[142] Hong Liu, Shuaipeng Zhang, Pengfei Zhang, Xinqiang Zhou, Xuebin Shao, Geguang Pu, and Yan Zhang. Blockchain and federated learning for collaborative intrusion detection in vehicular edge computing. *IEEE Transactions on Vehicular Technology*, 70(6):6073–6084, June 2021.

[143] Jing Liu and Zhentian Liu. A survey on security verification of blockchain smart contracts. *IEEE Access*, 7:77894–77904, 2019.

[144] Jinwei Liu, Haiying Shen, Husnu S Narman, Wingyan Chung, and Zongfang Lin. A survey of mobile crowdsensing techniques: A critical component for the internet of things. *ACM Transactions on Cyber-Physical Systems*, 2(3):1–26, 2018.

[145] Jinxin Liu, Burak Kantarci, and Carlisle Adams. Machine learning-driven intrusion detection for Contiki-NG-based iot networks exposed to NSL-KDD dataset. In *ACM Workshop on Wireless Security and ML*, pages 25–30, 2020.

[146] Jinxin Liu, Michele Nogueira, Johan Fernandes, and Burak Kantarci. Adversarial machine learning: A multilayer review of the state-of-the-art and challenges for wireless and mobile systems. *IEEE Communications Surveys Tutorials*, 24(1):123–159, 2022.

[147] Jinxin Liu, Yu Shen, Murat Simsek, Burak Kantarci, Hussein T. Mouftah, Mehran Bagheri, and Petar Djukic. A new realistic benchmark for advanced persistent threats in network traffic. *IEEE Networking Letters*, 4(3):162–166, 2022.

[148] Jinxin Liu, Murat Simsek, Burak Kantarci, Mehran Bagheri, and Petar Djukic. Collaborative Feature Maps of Networks and Hosts for AI-driven Intrusion Detection, August 2022. arXiv:2208.05085 [cs].

[149] Jinxin Liu, Murat Simsek, Burak Kantarci, Melike Erol-Kantarci, Andrew Malton, and Andrew Walenstein. Risk-aware fine-grained access control in cyber-physical contexts. *arXiv preprint arXiv:2108.12739*, 2021.

[150] Ming Liu, Zhi Xue, Xianghua Xu, Changmin Zhong, and Jinjun Chen. Host-based intrusion detection system with system calls: Review and future trends. *ACM Computing Surveys (CSUR)*, 51(5):1–36, 2018.

[151] Yuan Lu, Qiang Tang, and Guiling Wang. Zebralancer: Private and anonymous crowdsourcing system atop open blockchain. In *IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pages 853–865, 2018.

[152] Julián Luengo, Alberto Fernández, Salvador García, and Francisco Herrera. Addressing data complexity for imbalanced data sets: analysis of smote-based oversampling and evolutionary undersampling. *Soft Computing*, 15(10):1909–1936, Oct 2011.

[153] Zhihan Lv, Liang Qiao, Jinhua Li, and Houbing Song. Deep-learning-enabled security issues in the internet of things. *IEEE Internet of Things Journal*, 8(12):9531–9538, 2020.

[154] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[155] Andrzej Maćkiewicz and Waldemar Ratajczak. Principal components analysis (pca). *Computers & Geosciences*, 19(3):303–342, 1993.

[156] Mohammad Saeid Mahdavinejad, Mohammadreza Rezvan, Mohammadamin Barekatain, Peyman Adibi, Payam Barnaghi, and Amit P Sheth. Machine learning for internet of things data analysis: A survey. *Digital Communications and Networks*, 4(3):161–175, 2018.

[157] Florian Mendel, Norbert Pramstaller, Christian Rechberger, and Vincent Rijmen. Analysis of step-reduced sha-256. In *International workshop on fast software encryption*, pages 126–143. Springer, 2006.

[158] Yaroslav Meshcheryakov, Anna Melman, Oleg Evsutin, Vladimir Morozov, and Yevgeni Koucheryavy. On performance of pbft blockchain consensus algorithm for iot-applications with constrained devices. *IEEE Access*, 9:80559–80570, June 2021.

[159] Chenglin Miao, Qi Li, Houping Xiao, Wenjun Jiang, Mengdi Huai, and Lu Su. Towards data poisoning attacks in crowd sensing systems. In *Proceedings of the Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 111–120, 2018.

[160] Chenglin Miao, Qi Li, Houping Xiao, Wenjun Jiang, Mengdi Huai, and Lu Su. Towards data poisoning attacks in crowd sensing systems. In *ACM Intl. Symp. on Mobile Ad Hoc Networking and Computing*, Mobihoc '18, pages 111–120. ACM, 2018.

[161] Roweida Mohammed, Jumanah Rawashdeh, and Malak Abdullah. Machine learning with oversampling and undersampling techniques: overview study and experimental results. In *2020 11th international conference on information and communication systems (ICICS)*, pages 243–248. IEEE, 2020.

[162] Bhabendu Kumar Mohanta, Soumyashree S Panda, and Debasish Jena. An overview of smart contract and use cases in blockchain technology. In *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–4. IEEE.

[163] AH Mohsin, AA Zaidan, BB Zaidan, OS Albahri, AS Albahri, MA Alsalem, and KI Mohammed. Blockchain authentication of network applications: Taxonomy, classification, capabilities, open challenges, motivations, recommendations and future directions. *Computer Standards & Interfaces*, 64:41–60, 2019.

[164] Adel Ghazikhani ; Hadi Sadoghi Yazdi ; Reza Monsefi. Class imbalance handling using wrapper-based random oversampling. *20th Iranian Conference on Electrical Engineering (ICEE2012)*, 2012.

[165] Daesung Moon, Hyungjin Im, Ikkyun Kim, and Jong Hyuk Park. Dtb-ids: an intrusion detection system based on decision tree using behavior analysis for preventing apt attacks. *The Journal of supercomputing*, 73(7):2881–2895, 2017.

[166] Daesung Moon, Sung Bum Pan, and Ikkyun Kim. Host-based intrusion detection system for secure human-centric computing. *The Journal of Supercomputing*, 72(7):2520–2536, 2016.

[167] Alejandro Moreo, Andrea Esuli, and Fabrizio Sebastiani. Distributional random oversampling for imbalanced text classification. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 805–808, 2016.

[168] S. Morishima and H. Matsutani. Accelerating blockchain search of full nodes using gpus. In *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, pages 244–248.

[169] Nishat I Mowla, Nguyen H Tran, Inshil Doh, and Kijoon Chae. Federated learning-based cognitive detection of jamming attack in flying ad-hoc network. *IEEE Access*, 8:4338–4350, 2019.

[170] Noor Mualla, Essam H. Houssein, and Hala H. Zayed. Face age estimation approach based on deep learning and principle component analysis. *Intl. J. of Advanced Computer Science and Applications*, 9/2, 2018.

[171] Biswanath Mukherjee, L Todd Heberlein, and Karl N Levitt. Network intrusion detection. *IEEE network*, 8(3):26–41, 1994.

[172] Gervais Mwitende, Yalan Ye, Ikram Ali, and Fagen Li. Certificateless authenticated key agreement for blockchain-based wbans. *Journal of Systems Architecture*, page 101777, 2020.

[173] Marjan Naderan, Mehdi Dehghan, and Hossein Pedram. Sensing task assignment via sensor selection for maximum target coverage in wsns. *Journal of Network and Computer Applications*, 36(1):262–273, 2013.

[174] Dalia Nashat and Fatma A Hussain. Multifractal detrended fluctuation analysis based detection for syn flooding attack. *Computers & Security*, 107:102315, 2021.

[175] James Newsome, Elaine Shi, Dawn Song, and Adrian Perrig. The sybil attack in sensor networks: analysis & defenses. In *Third international symposium on information processing in sensor networks, 2004. IPSN 2004*, pages 259–268. IEEE.

[176] Sven Nõmm, Alejandro Guerra-Manzanares, and Hayretdin Bahsi. Towards the Integration of a Post-Hoc Interpretation Step into the Machine Learning Workflow for IoT Botnet Detection. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 1162–1169, December 2019.

[177] Joo-Hyuk Oh, Jae Yeol Hong, and Jun-Geol Baek. Oversampling method using outlier detectable generative adversarial network. *Expert Systems with Applications*, 133:1–8, 2019.

[178] Steve Omohundro. Cryptocurrencies, smart contracts, and artificial intelligence. *AI matters*, 1(2):19–21, 2014.

[179] O. Onireti, L. Zhang, and M. A. Imran. On the viable area of wireless practical byzantine fault tolerance (pbft) blockchain networks. In *IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, 2019.

[180] Ryan Osgood. The future of democracy: Blockchain voting. *COMP116: Information security*, pages 1–21, 2016.

[181] Safa Otoum, Burak Kantarci, and Hussein Mouftah. Empowering reinforcement learning on big sensed data for intrusion detection. In *Icc 2019-2019 IEEE international conference on communications (ICC)*, pages 1–7. IEEE, 2019.

[182] Safa Otoum, Burak Kantarci, and Hussein T. Mouftah. Detection of known and unknown intrusive sensor behavior in critical applications. *IEEE Sensors Letters*, 1(5):1–4, 2017.

[183] Safa Otoum, Burak Kantarci, and Hussein T Mouftah. Mitigating false negative intruder decisions in wsn-based smart grid monitoring. In *2017 13th International wireless communications and mobile computing conference (IWCMC)*, pages 153–158. IEEE, 2017.

[184] Safa Otoum, Burak Kantarci, and Hussein T Mouftah. On the feasibility of deep learning in sensor network intrusion detection. *IEEE Networking Letters*, 1(2):68–71, 2019.

[185] Safa Otoum, Burak Kantarci, and Hussein T. Mouftah. On the feasibility of deep learning in sensor network intrusion detection. *IEEE Networking Letters*, 1(2):68–71, 2019.

[186] Safa Otoum, Burak Kantarci, and Hussein 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. IEEE, 2020.

[187] Aafaf Ouaddah, Anas Abou Elkalam, and Abdellah Ait Ouahman. Towards a novel privacy-preserving access control model based on blockchain technology in iot. In Álvaro Rocha, Mohammed Serrhini, and Carlos Felgueiras, editors, *Europe and MENA Cooperation Advances in Information and Communication Technologies*, pages 523–533, Cham, 2017. Springer International Publishing.

[188] Mahesh Pal. Random forest classifier for remote sensing classification. *International journal of remote sensing*, 26(1):217–222, 2005.

[189] Maria Rita Palattella, Nicola Accettura, Luigi Alfredo Grieco, Gennaro Boggia, Mischa Dohler, and Thomas Engel. On optimal scheduling in duty-cycled industrial iot applications using ieee802. 15.4 e tsch. *IEEE Sensors Journal*, 13(10):3655–3666, 2013.

[190] Seong-Taek Park, Guozhong Li, and Jae-Chang Hong. A study on smart factory-based ambient intelligence context-aware intrusion detection system using machine learning. 11(4):1405–1412.

[191] Tao Peng, Kejian Guan, Jierong Liu, Jianer Chen, Guojun Wang, and Jiawei Zhu. A blockchain-based mobile crowdsensing scheme with enhanced privacy. *Concurrency and Computation: Practice and Experience*, page e6664, 2021.

[192] Husein Perez and Joseph HM Tah. Improving the accuracy of convolutional neural networks by identifying and removing outlier images in datasets using t-sne. *Mathematics*, 8(5):662, 2020.

[193] Muhammad Shakil Pervez and Dewan Md Farid. Feature selection and intrusion classification in nsl-kdd cup 99 dataset employing svms. In *The 8th International Conference on Software, Knowledge, Information Management and Applications (SKIMA 2014)*, pages 1–6. IEEE, 2014.

[194] Diah Anggraeni Pitaloka, Ajeng Wulandari, T Basaruddin, and Dewi Yanti Liliana. Enhancing cnn with preprocessing stage in automatic emotion recognition. *Procedia computer science*, 116:523–529, 2017.

[195] Mehrdokht Pournader, Yangyan Shi, Stefan Seuring, and SC Lenny Koh. Blockchain applications in supply chains, transport and logistics: a systematic review of the literature. *International Journal of Production Research*, 58(7):2063–2081, 2020.

[196] M. Pouryazdan, B. Kantarci, T. Soyata, and H. Song. Anchor-Assisted and Vote-based Trustworthiness Assurance in Smart City Crowdsensing. *IEEE Access*, 4:529–541, Mar. 2016.

[197] Maryam Pouryazdan, Claudio Fiandrino, Burak Kantarci, Tolga Soyata, Dzmitry Kliazovich, and Pascal Bouvry. Intelligent gaming for mobile crowd-sensing participants to acquire trustworthy big data in the internet of things. *IEEE Access*, 2017.

[198] Philipp Probst, Marvin N Wright, and Anne-Laure Boulesteix. Hyperparameters and tuning strategies for random forest. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 9(3):e1301, 2019.

[199] Alexandre Prud'Homme and Burak Kantarci. Poisoning attack anticipation in mobile crowdsensing: A competitive learning-based study. In *Proceedings of the 3rd ACM Workshop on Wireless Security and Machine Learning*, pages 73–78, 2021.

[200] Haytham Qushtom, Jelena Mišić, and Vojislav B Mišić. Multiple leader pbft based blockchain architecture for iot domains. In *2021 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 1–6. IEEE, 2021.

[201] Salman Rachmadi, Satria Mandala, and Dita Oktaria. Detection of dos attack using adaboost algorithm on iot system. In *2021 International Conference on Data Science and Its Applications (ICoDSA)*, pages 28–33. IEEE, 2021.

[202] José Ribeiro, Firooz B Saghezchi, Georgios Mantas, Jonathan Rodriguez, Simon J Shepherd, and Raed A Abd-Alhameed. An autonomous host-based intrusion detection system for android mobile devices. *Mobile Networks and Applications*, 25(1):164–172, 2020.

[203] David A Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang. Incremental learning for robust visual tracking. *International journal of computer vision*, 77(1):125–141, 2008.

[204] Ahmad-Reza Sadeghi, Christian Wachsmann, and Michael Waidner. Security and privacy challenges in industrial internet of things. In *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2015.

[205] Aldri L Santos, Christian AV Cervantes, Michele Nogueira, and Burak Kantarci. Clustering and reliability-driven mitigation of routing attacks in massive iot systems. *Journal of Internet Services and Applications*, 10(1):1–17, 2019.

[206] Azizah A. Manaf Mazdak Zamani Alireza Hooman Sasan Karamizadeh, Shahidan M. Abdullah. An overview of principal component analysis. *Journal of Signal and Information Processing*, 4:173–175, 2013.

[207] Robert E Schapire. Explaining adaboost. In *Empirical inference*, pages 37–52. Springer, 2013.

[208] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *International conference on artificial neural networks*, pages 583–588. Springer, 1997.

[209] Veronica Scuotto, Alberto Ferraris, Stefano Bresciani, Majed Al-Mashari, and Manlio Del Giudice. Internet of things: applications and challenges in smart cities. a case study of ibm smart city projects. *Business Process Management Journal*, 2016.

[210] Borja Seijo-Pardo, Iago Porto-Díaz, Verónica Bolón-Canedo, and Amparo Alonso-Betanzos. Ensemble feature selection: homogeneous and heterogeneous approaches. *Knowledge-Based Systems*, 118:124–139, February 2017.

[211] M Paz Sesmero, Agapito I Ledezma, and Araceli Sanchis. Generating ensembles of heterogeneous classifiers using stacked generalization. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 5(1):21–34, 2015.

[212] Hang Shen, Guangwei Bai, Yujia Hu, and Tianjing Wang. P2ta: Privacy-preserving task allocation for edge computing enhanced mobile crowdsensing. *Journal of Systems Architecture*, 97:130–141, 2019.

[213] Xiang Sheng, Jian Tang, and Weiyi Zhang. Energy-efficient collaborative sensing with mobile phones. In *INFOCOM, 2012 Proceedings IEEE*, pages 1916–1924. IEEE, 2012.

[214] Jimmy Shun and Heidar A. Malki. Network intrusion detection system using neural networks. In *2008 Fourth International Conference on Natural Computation*, volume 5, pages 242–246, 2008.

[215] Daniel L Silver, Qiang Yang, and Lianghao Li. Lifelong machine learning systems: Beyond learning algorithms. In *2013 AAAI spring symposium series*, 2013.

[216] Pradeep Singh and M Venkatesan. Hybrid approach for intrusion detection system. In *2018 International Conference on Current Trends towards Converging Technologies (ICCTCT)*, pages 1–5. IEEE, 2018.

[217] A. Sood, M. Simsek, Y. Zhang, and B. Kantarci. Deep learning-based detection of fake task injection in mobile crowdsensing. In *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 1–5, 2019.

[218] A. Sood, M. Simsek, Y. Zhang, and B. Kantarci. Deep learning-based detection of fake task injection in mobile crowdsensing. In *IEEE Global Conf. on Signal and Information Processing*, Nov. 2019 (accepted).

[219] Ployphan Sornsuwit and Saichon Jaiyen. Intrusion detection model based on ensemble learning for u2r and r2l attacks. In *2015 7th international conference on information technology and electrical engineering (ICITEE)*, pages 354–359. IEEE, 2015.

[220] Deris Stiawan, Mohd Yazid Bin Idris, Alwi M Bamhdi, Rahmat Budiarto, et al. Cicids-2017 dataset feature analysis with information gain for anomaly detection. *IEEE Access*, 8:132911–132921, 2020.

[221] Biljana L Risteska Stojkoska and Kire V Trivodaliev. A review of internet of things for smart home: Challenges and solutions. *Journal of Cleaner Production*, 140:1454–1464, 2017.

[222] Hongjun Su, Yao Yu, Qian Du, and Peijun Du. Ensemble learning for hyperspectral image classification using tangent collaborative representation. *IEEE Transactions on Geoscience and Remote Sensing*, 2020.

[223] Tongtong Su, Huazhi Sun, Jinqi Zhu, Sheng Wang, and Yabo Li. Bat: deep learning methods on network intrusion detection using nsl-kdd dataset. *IEEE Access*, 8:29575–29585, 2020.

[224] Abid Sultan, Muhammad Azhar Mushtaq, and Muhammad Abubakar. Iot security issues via blockchain: A review paper. In *Proceedings of the 2019 International Conference on Blockchain Technology*, ICBCT 2019, page 60–65, New York, NY, USA. Association for Computing Machinery.

[225] Nasrin Sultana, Naveen Chilamkurti, Wei Peng, and Rabei Alhadad. Survey on sdn based network intrusion detection system using machine learning approaches. *Peer-to-Peer Networking and Applications*, 12(2):493–501, 2019.

[226] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification? In *China National Conference on Chinese Computational Linguistics*, pages 194–206. Springer, 2019.

[227] Bayu Adhi Tama and Sunghoon Lim. Ensemble learning for intrusion detection systems: A systematic mapping study and cross-benchmark evaluation. *Computer Science Review*, 39:100357, 2021.

[228] Xi Tao and Abdelhakim Senhaji Hafid. Chainsensing: A novel mobile crowdsensing framework with blockchain. *IEEE Internet of Things Journal*, 9:2999–3010, February 2021.

[229] N. Tapas, G. Merlino, and F. Longo. Blockchain-based iot-cloud authorization and delegation. In *IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 411–416, 2018.

[230] Saurabh Tewari and UD Dwivedi. A comparative study of heterogeneous ensemble methods for the identification of geological lithofacies. *Journal of Petroleum Exploration and Production Technology*, pages 1–20, 2020.

[231] Evangelos Theodoridis, Georgios Mylonas, and Ioannis Chatzigiannakis. Developing an iot smart city framework. In *IISA 2013*, pages 1–6. IEEE, 2013.

[232] Richard Lee Twesige. A simple explanation of bitcoin and blockchain technology, 2015.

[233] John R Vacca. *Network and system security*. Elsevier, 2013.

[234] Abhishek Verma and Virender Ranga. Machine learning based intrusion detection systems for IoT applications. 111(4):2287–2310.

[235] D. Vujicic, D. Jagodic, and S. Ranić. Blockchain technology, bitcoin, and ethereum: A brief overview. In *2018 17th International Symposium INFOTEH-JAHORINA (INFOTEH)*, pages 1–6, March.

[236] Arfan Haider Wahla, Lan Chen, Yali Wang, Rong Chen, and Fan Wu. Automatic wireless signal classification in multimedia internet of things: An adaptive boosting enabled approach. *IEEE Access*, 7:160334–160344, 2019.

[237] Xiaoyue Wan, Geyi Sheng, Yanda Li, Liang Xiao, and Xiaojiang Du. Reinforcement learning based mobile offloading for cloud-based malware detection. In *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pages 1–6. IEEE, 2017.

[238] Chao-Ran Wang and Xin-Hui Shao. An improving majority weighted minority oversampling technique for imbalanced classification problem. *IEEE Access*, 9:5069–5082, 2020.

[239] Haohan Wang and Bhiksha Raj. On the origin of deep learning. *arXiv preprint arXiv:1702.07800*, 2017.

[240] Hui Wang, Ivo Duentsch, Gongde Guo, and Sadiq Ali Khan. Special issue on small data analytics. *International Journal of Machine Learning and Cybernetics*, 14(1):1–2, 2023.

[241] Jiangtao Wang, Leye Wang, Yasha Wang, Daqing Zhang, and Linghe Kong. Task allocation in mobile crowd sensing: State-of-the-art and future opportunities. *IEEE Internet of Things journal*, 5(5):3747–3757, 2018.

[242] Leye Wang, Daqing Zhang, Dingqi Yang, Brian Y Lim, and Xiaojuan Ma. Differential location privacy for sparse mobile crowdsensing. In *IEEE 16th International Conference on Data Mining (ICDM)*, pages 1257–1262, 2016.

[243] Shen Wang, Ahmad F Taha, and Jianhui Wang. Blockchain-assisted crowdsourced energy systems. In *IEEE Power & Energy Society General Meeting (PESGM)*, pages 1–5, 2018.

[244] Xiali Wang and Xiang Lu. A host-based anomaly detection framework using xgboost and lstm for iot devices. *Wireless Communications and Mobile Computing*, 2020, 2020.

[245] Y. Wang and B. Kantarci. A novel reputation-aware client selection scheme for federated learning within mobile environments. In *IEEE CAMAD*, pages 1–6, 2020.

[246] Yanan Wang, Guodong Sun, and Xingjian Ding. Coverage-balancing user selection in mobile crowd sensing with budget constraint. *Sensors*, 19/10:2371, 2019.

[247] Aaron Wright and Primavera De Filippi. Decentralized blockchain technology and the rise of lex cryptographia, 2015. *Retrieved August*, 1:2017, 2015.

[248] Hao-Tian Wu, Yucong Zheng, Bowen Zhao, and Jiankun Hu. An anonymous reputation management system for mobile crowdsensing based on dual blockchain. *IEEE Internet of Things Journal*, 9/9:6956–6968, Sep. 2021.

[249] L. Xiao, D. Jiang, D. Xu, W. Su, N. An, and D. Wang. Secure mobile crowdsensing based on deep learning. *China Communications*, 15(10):1–11, Oct. 2018.

[250] L. Xiao, Y. Li, G. Han, H. Dai, and H. V. Poor. A secure mobile crowdsensing game with deep reinforcement learning. *IEEE Transactions on Information Forensics and Security*, 13(1):35–47, Jan. 2018.

[251] Liang Xiao, Donghua Jiang, Dongjin Xu, and Ning An. Secure mobile crowdsensing with deep learning, arXiv, eprint: 1801.07379, 2018.

[252] Liang Xiao, Yanda Li, Guoan Han, Huaiyu Dai, and H Vincent Poor. A secure mobile crowdsensing game with deep reinforcement learning. *IEEE Transactions on Information Forensics and Security*, 13(1):35–47, 2017.

[253] Haoyi Xiong, Daqing Zhang, Leye Wang, and Hakima Chaouchi. Emc 3: Energy-efficient data transfer in mobile crowdsensing under full coverage constraint. *IEEE Transactions on Mobile Computing*, 14(7):1355–1368, 2014.

[254] Jinbo Xiong, Xiuhua Chen, Qing Yang, Lei Chen, and Zhiqiang Yao. A task-oriented user selection incentive mechanism in edge-aided mobile crowdsensing. *IEEE Transactions on Network Science and Engineering*, 7(4):2347–2360, 2019.

[255] Guowen Xu, Hongwei Li, Sen Liu, Kan Yang, and Xiaodong Lin. Verifynet: Secure and verifiable federated learning. *IEEE Transactions on Information Forensics and Security*, 15:911–926, 2019.

[256] Zheng Xu, Chaofan Liu, Peng Zhang, Tun Lu, and Ning Gu. Urim: Utility-oriented role-centric incentive mechanism design for blockchain-based crowdsensing. In *International Conference on Database Systems for Advanced Applications*, volume 12683, pages 358–374. Springer, April 2021.

[257] Ke Yan, Guoming Lu, Guangchun Luo, Xu Zheng, Ling Tian, and Akshita Maradapu Vera Venkata Sai. Location privacy-aware task bidding and assignment for mobile crowd-sensing. *IEEE Access*, 7:131929–131943, 2019.

[258] Ke Yan, Guangchun Luo, Xu Zheng, Ling Tian, and Akshita Maradapu Vera Venkata Sai. A comprehensive location-privacy-awareness task selection mechanism in mobile crowd-sensing. *IEEE Access*, 7:77541–77554, 2019.

[259] Mengmeng Yang, Tianqing Zhu, Kaitai Liang, Wanlei Zhou, and Robert H. Deng. A blockchain-based location privacy-preserving crowdsensing system. *Future Generation Computer Systems*, 94:408 – 418, 2019.

[260] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.

[261] Xiong Yang, Yuling Chen, Xiaobin Qian, Tao Li, and Xiao Lv. Bcead: A blockchain-empowered ensemble anomaly detection for wireless sensor network via isolation forest. *Security and Communication Networks*, 2021:1–10, Nov 2021.

[262] Xun Yi, Kwok-Yan Lam, Elisa Bertino, and Fang-Yu Rao. Location privacy-preserving mobile crowd sensing with anonymous reputation. In *European Symposium on Research in Computer Security*, pages 387–411. Springer, 2019.

[263] Chuanlong Yin, Yuefei Zhu, Jinlong Fei, and Xinzheng He. A deep learning approach for intrusion detection using recurrent neural networks. *IEEE Access*, 5:21954–21961, 2017.

[264] Lean Yu, Rongtian Zhou, Ling Tang, and Rongda Chen. A dbn-based resampling svm ensemble learning paradigm for credit classification with imbalanced data. *Applied Soft Computing*, 69:192 – 202, 2018.

[265] Yufeng Zhan, Peng Li, Zhihao Qu, Deze Zeng, and Song Guo. A learning-based incentive mechanism for federated learning. *IEEE Internet of Things Journal*, 2020.

[266] C. Zhang, K. C. Tan, and R. Ren. Training cost-sensitive deep belief networks on imbalance data problems. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 4362–4367, July 2016.

[267] C. Zhang, L. Zhu, C. Xu, X. Liu, and K. Sharif. Reliable and privacy-preserving truth discovery for mobile crowdsensing systems. *IEEE Transactions on Dependable and Secure Computing*, pages 1–1, 2019.

[268] Chongzhen Zhang, Yanli Chen, Yang Meng, Fangming Ruan, Runze Chen, Yidan Li, and Yaru Yang. A novel framework design of network intrusion detection based on machine learning techniques. *Security and Communication Networks*, 2021, 2021.

[269] Min Zhang, Tao Yu, and Guo Fang Zhai. Smart transport system based on "the internet of things". In *Applied mechanics and materials*, volume 48, pages 1073–1076. Trans Tech Publ, 2011.

[270] Y. Zhang, M. Simsek, and B. Kantarci. Self organizing feature map for fake task attack modelling in mobile crowdsensing. In *IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, 2019.

[271] Yanping Zhang, Yang Xiao, Kaveh Ghaboosi, Jingyuan Zhang, and Hongmei Deng. A survey of cyber crimes. *Security and Communication Networks*, 5(4):422–437, 2012.

[272] Yueqian Zhang and Burak Kantarci. AI-based security design of mobile crowdsensing systems: Review, challenges and case studies. In *IEEE Intl Conf on Service-Oriented System Engineering (SOSE)*, pages 17–26, 2019.

[273] Yueqian Zhang and Burak Kantarci. Ai-based security design of mobile crowdsensing systems: Review, challenges and case studies. In *IEEE International Conference on Service-Oriented System Engineering (SOSE)*, pages 17–1709, 2019.

[274] Yueqian Zhang, Murat Simsek, and Burak Kantarci. Machine learning-based prevention of battery-oriented illegitimate task injection in mobile crowdsensing. In *Proceedings of the ACM Workshop on Wireless Security and Machine Learning*, WiseML 2019, pages 31–36. ACM, 2019.

[275] Yueqian Zhang, Murat Simsek, and Burak Kantarci. Self organizing feature map for fake task attack modelling in mobile crowdsensing. In *Global Communications Conference (GLOBECOM), 2019*. IEEE (Accepted), Preprint available: http://nextconlab.academy/Zhang-GC19.zip ; Pwd:ACMTCPS, 2019.

[276] Hui Zhao, Mingjun Xiao, Jie Wu, Yun Xu, He Huang, and Sheng Zhang. Differentially private unknown worker recruitment for mobile crowdsensing using multi-armed bandits. *IEEE Transactions on Mobile Computing*, pages 1–1, 2020.

[277] Ke Zhao, Shaohua Tang, Bowen Zhao, and Yiming Wu. Dynamic and privacy-preserving reputation management for blockchain-based mobile crowdsensing. *IEEE Access*, 7:74694–74710, 2019.

[278] Mengchen Zhao, Bo An, Yaodong Yu, Sulin Liu, and Sinno Jialin Pan. Data poisoning attacks on multi-task relationship learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[279] Zibin Zheng, Shaoan Xie, Hongning Dai, Xiangping Chen, and Huaimin Wang. An overview of blockchain technology: Architecture, consensus, and future trends. In *IEEE international congress on big data (BigData congress)*, pages 557–564, 2017.

[280] Qiheng Zhou, Huawei Huang, Zibin Zheng, and Jing Bian. Solutions to scalability of blockchain: A survey. *Ieee Access*, 8:16440–16455, 2020.

[281] Tongqing Zhou, Zhiping Cai, Kui Wu, Yueyue Chen, and Ming Xu. Fidc: A framework for improving data credibility in mobile crowdsensing. *Computer Networks*, 120:157–169, 2017.

[282] Zhi-Hua Zhou. Ensemble learning. In *Machine Learning*, pages 181–210. Springer, 2021.

[283] Shihong Zou, Jinwen Xi, Honggang Wang, and Guoai Xu. Crowdblps: A blockchain-based location-privacy-preserving mobile crowdsensing system. *IEEE Transactions on Industrial Informatics*, 16(6):4206–4218, 2019.