DEMOCRATIC AND POPULAR REPUBLIC OF ALGERIA

MINISTRY OF HIGHER EDUCATION AND RESEARCH

SAAD DAHLAB UNIVERSITY, BLIDA 1

Faculty of Technology

Electronics Department

LATSI Laboratory

DOCTORAL THESIS OF THE THIRD CYCLE

Option: Embedded Systems

APPLICATION OF ARTIFICIAL INTELLIGENCE FOR IOT DATA ANALYSIS

By

Abdelkader HADJ-ATTOU

In front of the jury:

Zoubir BENSELAMA	Professor, University of Blida 1	President
Merouane MEHDI	MC-A, University of Blida 1	Examiner
Abdessalem BENAMMAR	Research Director, CRTI-Cheraga	Examiner
Farid YKHLEF	Professor, University of Blida 1	Supervisor
Yacine KABIR	MA-A, University of Blida 1	Invited

Blida, December, 2024

ABSTRACT

The main objective of this thesis is to use artificial intelligence to support the large number of IoT devices used in fast-growing fields such as automotive and healthcare. We will explore how artificial intelligence can address some of the limitations of IoT technology and how it can boost the deployment of IoT technology in human lives. This is useful in existing automated monitoring systems to provide comprehensive, real-time, reliable, and automated IoT monitoring based on artificial intelligence solutions. It is a strategic area for IoT in general and embedded systems, but more specifically for autonomous vehicles, healthcare, and smart cities. Another objective is to carry out studies and analyses on IoT in a Big Data context. Additionally, we will provide an AI-based security method that automatically includes the growing security aspects related to the IoT domain. In this thesis, we propose four frameworks that classify IoT data by utilizing hybrid deep learning models and ensemble learning methods. Two road datasets have been created and labeled in real-time, providing the necessary resources for the evaluation of our proposed models. The first three frameworks demonstrate the effectiveness of hybrid models and ensemble techniques in improving road surface anomaly detection. Also, we propose a big data framework that employs Apache Spark and ensemble learning to improve IoT network intrusion detection. The fourth framework passes through extensive testing using the BoT-IoT dataset, proving its ability to handle traffic imbalances and detect various IoT network threats with high accuracy. The experimental results clearly show the effectiveness and reliability of the four frameworks.

Keywords: Artificial Intelligence, Big Data, IoT Technology, IoT Security.

RÉSUMÉ

L'objectif principal de cette thèse est d'utiliser l'intelligence artificielle pour prendre en charge le grand nombre d'appareils IoT utilisés dans les domaines en forte croissance tel que l'automobile et la santé. Nous étudierons comment l'intelligence artificielle peut résoudre certaines limitations de la technologie IoT et comment elle peut stimuler le déploiement de la technologie IoT dans la vie de l'homme. Ceci est utile dans les systèmes de surveillance automatisé existant afin de fournir une surveillance complète en temps réel, fiable et automatisée des IoT basée sur des solutions d'intelligence artificielle. C'est un domaine stratégique pour les IoT en général et les systèmes embarqués, mais plus particulièrement pour les véhicules autonomes, les soins de santé et les villes intelligentes. Un autre objectif est de réaliser des études et analyses sur l'IoT dans un contexte Big Data. En plus, nous allons proposer une méthode de sécurité basée sur l'intelligence artificielle qui inclut automatiquement les aspects de sécurité croissants liés au domaine de l'IoT. Dans cette thèse, nous proposons quatre structures visant à classer les données IoT en utilisant des modèles hybrides d'apprentissage profond et des méthodes d'apprentissage ensembliste. Pour évaluer nos modèles, deux ensembles de données routières ont été créés et étiquetés en temps réel, fournissant les ressources nécessaires à l'évaluation de nos modèles proposés. Les trois premières structures démontrent l'efficacité des modèles hybrides et des techniques ensemblistes pour améliorer la détection des anomalies sur les surfaces routières. En outre, nous proposons un structure de big data qui utilise Apache Spark et l'apprentissage ensembliste pour renforcer la détection des intrusions dans les réseaux IoT. La quatrième structure passe par des tests approfondis en utilisant l'ensemble de données BoT-IoT, prouvant sa capacité à gérer les déséquilibres de trafic et à détecter diverses menaces sur le réseau IoT avec une grande précision. Les résultats expérimentaux confirment l'efficacité et la fiabilité des quatre structures proposés.

Mots clés : Intelligence Artificielle, Big Data, Technologie IoT, Sécurité IoT.

الملخص

الهدف الرئيسي من هذه الأطروحة هو استخدام الذكاء الاصطناعي لدعم العدد الكبير من أجهزة إنترنت الأشياء المستخدمة في المجالات سريعة النمو مثل السيارات والرعاية الصحية. سنستكشف كيف يمكن للذكاء الاصطناعي أن يعالج بعض القيود المفروضة على تكنولوجيا إنترنت الأشياء وكيف يمكن أن يعزز نشر تكنولوجيا إنترنت الأشياء في حياة الإنسان. وهذا مفيد في أنظمة المراقبة الآلية الحالية لتوفير مراقبة إنترنت الأشياء الشاملة والموثوقة والآلية في الوقت الفعلي بناءً على حلول الذكاء الاصطناعي. يعد هذا مجالًا استراتيجيًا لإنترنت الأشياء بشكل عام والأنظمة المدمجة، ولكن بشكل أكثر تحديدًا للمركبات ذاتية القيادة والرعاية الصحية والمدن الذكية. الهدف الآخر هو إجراء دراسات وتحليلات حول إنترنت الأشياء في سياق البيانات الضخمة. بالإضافة إلى ذلك، سنوفر طريقة أمان قائمة على الذكاء الاصطناعي تتضمن تلقائيًا الجوانب الأمنية المتنامية المتعلقة بمجال إنترنت الأشياء. في هذه الأطروحة، نقترح أربعة أطر عمل لتصنيف بيانات إنترنت الأشياء من خلال الاستفادة من نماذج التعلم العميق الهجينة وطرق التعلم الجماعي. تم إنشاء مجموعتين من بيانات الطرق ووضع علامات عليها في الوقت الفعلي، مما يوفر الموارد اللازمة لتقييم النماذج المقترحة. توضح الأطر الثلاثة الأولى فعالية النماذج الهجينة وتقنيات التعلم الجماعي في تحسين اكتشاف شذوذ سطح الطريق. كما نقترح إطار عمل للبيانات الضخمة يستخدم Apache Spark والتعلم الجماعي لتحسين اكتشاف اختراق شبكة إنترنت الأشياء. يمر الإطار الرابع باختبارات مكثفة باستخدام مجموعة بيانات BoT-IoT، مما يثبت قدرته على التعامل مع اختلالات حركة المرور واكتشاف تهديدات شبكة إنترنت الأشياء المختلفة بدقة عالية. تظهر النتائج التجريبية بوضوح فعالية وموثوقية الأطر الأربعة.

الكلمات المفتاحية : الذكاء الاصطناعي، البيانات الضخمة، تكنولوجيا إنترنت الأشياء، أمن إنترنت الأشياء.

DEDICATIONS

I dedicate my thesis work to my wonderful parents, for all their sacrifices, love, support, and prayers during my studies,

To my professors at the Electronics Department of Saad Dahlab University,

To my family and friends for their support during my university career,

To the challenges that made me stronger,

ACKNOWLEDGEMENTS

In the name of Allah, the Most Gracious and Merciful, Alhamdulillah, thanks to Allah for the strength and blessing in completing this thesis.

I have had a lot of help and support while I have been writing my thesis. Thus, with these few lines, I would like to express my gratitude to everyone who helped to complete my thesis, whether directly or indirectly.

I would like to express my sincere gratitude to my supervisor, Farid Ykhlef, a professor at the University of Blida 1. He directed my thesis, kindly helped me along every step, and gave me a foundation that was really useful.

I am grateful to Mr. Yacine Kabir, a professor at Blida University, for his help over the past six years. You were incredibly approachable, gave excellent research suggestions, and had a lot of patience in teaching me how to establish ideas and gain an understanding of the work we were doing.

Also, I extend my heartfelt gratitude to the entire LATSI Laboratory team, I wish you luck in your future projects.

Lastly, I just wanted to thank everyone again for helping me to complete this thesis. The final aim is that others may find this research valuable.

Abdelkader Hadj-Attou

2024

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

IoT	Internet of Things
AI	Artificial Intelligence
IDS	Intrusion Detection Systems
ML	Machine Learning
DL	Deep Learning
M2M	Machine-to-Machine
IIoT	Industrial Internet of Things
GPIO	General Purpose Input Output
LAN	Local Area Network
LTE	Long-Term Evolution
LPWAN	Low Power Wide Area Network
BLE	Bluetooth Low Energy
WiFi	Wireless Fidelity
WLAN	Wireless Local Area Network
HTTP	Hypertext Transfer Protocol
XMPP	Extensible Messaging and Presence Protocol
XML	eXtensible Markup Language
TCP	Transmission Control Protocol
MQTT	Message Queuing Telemetry Transport
IETF	Internet Engineering Task Force
W3C	World Wide Web Consortium
XSF	XMPP Standards Foundation
SQL	Structured Query Language
DBMS	Database Management System
BSON	Binary Javascript Object Notation
StaaS	Storage as a Service
SaaS	Software as a Service
PaaS	Platform as a Service
IaaS	Infrastructure as a Service
AIoT	Agricultural IoT
DT	Decision Tree

RF	Random Forest
KNN	K-nearest Neighbor
SVM	Support Vector Machines
NB	Naive Bayes
RL	Reinforcement Learning
DNN	Deep Neural Network
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
GRU	Gated Recurrent Unit
EL	Ensemble Learning
GBM	Gradient Boosting Machine
XGBM	Extreme Gradient Boosting Machine
ITS	Intelligent Transportation Systems
RSC	Road Surface Condition
DTW	Dynamic Time Warping
GMM	Gaussian Mixture Model
IMU	Inertial Measurement Unit
GPS	Global Positioning System
FFT	Fast Fourier Transform
PSD	Power Spectral Density
DWT	Discrete Wavelet Transform
CWT	Continuous Wavelet Transform
HAR	Human Activity Recognition
TD	TimeDistributed
SMOTE	Synthetic Minority Oversampling Technique
GAN	Generative Adversarial Network
RDD	Resilient Distributed Dataset
DDoS	Distributed Denial of Service
DoS	Denial of Service

GENERAL INTRODUCTION

The Internet of Things (IoT) involves most physical things or devices that can connect to the Internet, including manufacturing equipment and machinery, household items, and automobiles. Specifically, the concept of IoT today refers to a network of networked devices that share data with other devices and incorporate software, sensors, and processing power. The Internet and the IoT are two different concepts. It would be reasonable to say that the Internet of Things is more intelligent than the Internet since it has the ability to generate, evaluate, and make decisions based on data from connected devices. IoT is changing numerous applications, and the data created by IoT devices has the potential to significantly enhance company profits after the analysis of the data [1, 2].

Through the use of modern wireless technologies like actuators, sensors, and smartphones, businesses embrace IoT to incorporate every task and assist in achieving their objectives. The implementation of IoT benefits businesses, since it can reinforce a company strategy based on technology and social networks. The current and next generation of IoT technologies have significant potential for enhancing the overall quality of people's lives through automation, productivity, and user convenience over a variety of application fields [3]. Smart cities, smart industries, smart transportation, healthcare, and water monitoring are some of the best examples of IoT applications.

Any object in the world with the technological features needed to allow it to connect to the Internet via a wireless or wired connection can be considered an IoT device. Users of IoT may be machines, people, or both. IoT is a combination of various technologies that enable the networking of many things, not a single device or technology [4]. IoT collects data from a variety of sources, including environmental sensors, smart devices, and smart cars. After that, the data can be transferred to the edge gateway and cloud via the internet using widely used standard protocols like HTTP, XMPP, WebSocket, and MQTT. Functionalities like preprocessing the data, protecting cloud connectivity, and collecting sensor data are all provided by the edge gateway. Several database management systems designed for IoT applications are available in the cloud. To provide meaningful information, the data are analyzed using artificial intelligence, and fundamental computing methods. With

the usage of this data, businesses may improve activities, interact with consumers more, automate process control, and make the best decisions possible based on the insights obtained from the data [5].

Artificial intelligence (AI) is the term used to describe the imitation of human intelligence in robots that are designed to think and behave like people. Machines with AI are able to learn and respond to a variety of scenarios and problems. Due to its enormous flexibility, AI can be used in combination with the IoT. For instance, smart home devices or microcontroller chips can form several IoT networks to collect, analyze, and exchange various types of data flows over the internet [1, 2].

The exponential increase in the amount of data transferred between IoT devices is the main obstacle in implementing IoT systems. Moreover, a thorough analysis of all this data is required. IoT data is consequently analyzed using a large number of big data methods. Since the data collected by IoT devices is unstructured, big data uses multiple storage approaches to analyze and store the collected data in real time. These massive amounts of data are known as big data when artificial intelligence is employed to provide useful information [5-7].

The main concern when implementing IoT technology is security. IoT devices that do not have protection pose a risk to the overall IoT infrastructure. In general, the processing capacity of IoT devices is limited. Due to these limitations, they are unable to secure their information as well as interact with other devices using fundamental security features like firewalls or robust cryptosystems. Numerous factors contribute to IoT safety concerns, including a lack of hardware safety features, poorly written software with wide limits on vulnerabilities, and various errors in safety design [4, 8].

Research Contribution

The main goal of this thesis is to use AI to assist the enormous number of IoT devices employed in the field of IoT. The first phase includes collecting IoT data and sending it to a server for storage. In the second phase, data preprocessing is utilized to reduce the amount of computation required by cleaning, transforming, and organizing the data to boost the reliability of the final decision. The final part is an AI-based classification phase that uses

machine learning (ML), deep learning (DL), hybrid deep learning, and ensemble learning algorithms.

Artificial intelligence is a critical field for embedded systems and the IoT in general, but especially for smart cities, healthcare, and self-driving cars. The ability to use IoT sensor data is getting a lot of attention in this field. The data will become incredibly large, so we will need to adopt big data solutions. This thesis also looks into how AI might help overcome the security concerns associated with IoT technology in order to expand its acceptance in daily life. Our solution includes four frameworks designed to enhance the precision and reliability of IoT data analysis. Three frameworks are focused on monitoring road surface conditions, which detect road anomalies by analyzing vibrations from motion sensors in moving vehicles, and one is designed to detect IoT network attacks. In order to classify road surface anomalies, the first proposed framework employed smartphone motion sensors to create a new road dataset. Then, hybrid deep learning models were designed to classify road surface anomalies. The second framework included another road dataset collected using an IoT device to enable the proposed 3D hybrid deep learning models for identifying road surface conditions. For further improvement of 3D hybrid deep learning models, the third framework employed data augmentation techniques on the road dataset. The efficiency of these techniques was investigated through the application of ensemble learning methods. Finally, the fourth framework implemented a machine learning approach within a Big Data environment to detect intrusions in the BoT-IoT dataset. It effectively addressed data imbalance challenges by integrating oversampling techniques with ensemble resampling methods. The following briefly describes the primary contributions of our work:

- Create road datasets to monitor road surface conditions based on motion sensors embedded in IoT devices or smartphones. The problem of class imbalance in road data is addressed via data augmentation techniques.
- Develop hybrid deep learning models for classifying road surface anomalies. We also evaluate the efficacy of using spatiotemporal features as inputs.
- Develop a Big Data framework based on Apache Spark for network traffic monitoring. The methodology uses an ensemble learning approach and an oversampling technique to improve classification performance and address the issue of class imbalance.

Thesis Structure

There are four chapters in this thesis, arranged as follows:

In the first chapter, we will provide a description of the protocols, applications, and architectures of the Internet of Things. We will also discuss various AI approaches, IoT security, and big data solutions.

The second chapter discusses some common techniques for road data analysis and road surface conditions monitoring. In addition, we will provide IoT-based architectures for monitoring road surface conditions, as well as AI models for detecting road anomalies.

The third chapter will be devoted on creating a big data architecture using Apache Spark to identify IoT network intrusions. In this chapter, we will investigate how ensemble learning methods and machine learning approaches can help to enhance the accuracy of IoT network intrusion detection.

The fourth chapter will cover the experiments conducted to evaluate IoT data using AI algorithms. In these experiments, various hybrid deep learning models that were created for the classification of road surface anomalies are evaluated. Furthermore, the BoT-IoT dataset will be used in the intrusion detection experiments.

List of Publications

- **Journal paper:** Hadj-Attou, Abdelkader, Yacine Kabir, and Farid Ykhlef. "Hybrid deep learning models for road surface condition monitoring." Measurement 220 (2023): 113267.
- **Conference paper:** Hadj-Attou, Abdelkader, Yacine Kabir, and Farid Ykhlef. "IoT Based Road Surface Condition Monitoring Using Spatiotemporal Feature Learning." International Conference on Computing Systems and Applications. Cham: Springer Nature Switzerland, 2024.
- **Conference paper:** Hadj-Attou, Abdelkader, Yacine Kabir, and Farid Ykhlef. "A Big Data Security Framework for IoT Networks using Weighted Average Ensemble." 2024 2nd International Conference on Electrical Engineering and Automatic Control (ICEEAC). IEEE, 2024.

CHAPTER 1: STATE OF THE ART

1.1 Introduction

Modern technological advancements have improved and facilitated people's lives while offering numerous advantages. This progress is seen in the community, medical, academic, and economic areas among others. These days, the Internet is nearly everywhere, has impacted almost every region of the world, and is having unthinkable effects on human life. However, a very wide range of devices can be connected to the Internet, resulting in an era of everywhere connectivity called the Internet of Things (IoT). The term "IoT" describes the widespread smart devices equipped with Internet connections [5, 9].

There is an ongoing increase in the quantity of IoT devices. The total amount of IoT devices is predicted to increase from 500 million in 2003 to 24.1 billion by the end of 2030, representing approximately 3.47 IoT devices for each user. Around 5.8 billion of the 24.1 billion devices are expected to be reserved for business and industrial use. These facts demonstrate the significance of IoT since it indicates how individuals and devices are fundamentally changing the way connected environments are measured, sensed, and communicated with [10].

IoT devices have embedded transceivers, CPUs, actuators, and sensors to enable this intelligence and connectivity. IoT is a combination of multiple technologies that operate in harmony rather than being a single technology. Devices that provide interaction with the physical world include sensors and actuators. To extract significant information from the sensor data, it needs to be intelligently stored and analyzed [5].

IoT has become a popular topic in news articles and marketing trends. Sensor networks, embedded systems, and computer science are some of the earlier techniques that gave rise to IoT. In addition, IoT emerged as an effective way to interact with machines across many areas. Although a lot of IoT devices are networked together to create systems for specialized purposes, they rarely serve as public access devices in the global internet [7].

In IoT applications, artificial intelligence (AI) is becoming more and more important. The ability of AI to swiftly extract insights from data is what makes it valuable in the IoT field. The IoT environment can benefit from the use of AI in a number of ways to improve

its efficiency and create more intelligent and effective IoT applications. Indeed, AI algorithms enable the identification of patterns in data that might be challenging for humans to recognize. This can be helpful in identifying anomalies, allowing companies to make better decisions [11-13].

Like all technology, IoT has been predicted to have both positive and negative effects on modern life. Advocates of the Internet of Things argue that it represents the first true advancement in the Internet and could improve people's lives in a variety of ways, including employment, education, and entertainment. The biggest issues with IoT in terms of disadvantages are security and privacy risks. Devices connected to the Internet of Things produce, transfer, and preserve important data on the routines, pursuits, traits, and personalities of their users as well as details about their immediate surroundings. The issue at hand is that IoT devices are vulnerable to security flaws, and privacy rules are unable to mitigate the risks posed by this growing problem [9].

In this chapter, we give an overview of IoT architecture, applications, and protocols. We also cover big data, IoT security systems, and potential AI techniques.

1.2 Internet of Things (IoT)

The integration of communication features into everyday objects and utilizing new Internet technologies created the Internet of Things (IoT). This revolutionary networking framework that is called IoT enables communication over the Internet between various physical devices [14]. The term IoT describes an interconnected system of smart devices that are capable of collecting and sharing data since they include internet connection, software, and sensors. Almost every physical thing that has the ability to be connected to the internet and transmit data has the potential to become an IoT device. These devices may include anything from connected streetlights, smart thermostats, driverless trucks, and children's toys [15, 16]. The IoT system's goal is to enhance people's quality of life through allowing people to respond effectively to changes in the environment and by offering services that are customized for the individual needs of users [4].

In the 1980s and 1990s, there was discussion about integrating sensors and intelligence into everyday things. Although there were a few initial initiatives, such as a vending machine

linked to the Internet, advancement remained slow due to the unavailability of necessary technologies [16]. At first, the manufacturing and business sectors were the areas where the IoT was the most fascinating due to its machine-to-machine (M2M) applications. As a subset of the IoT, M2M solutions employ wireless technologies to connect devices to the Internet and each other, requiring limited human involvement for providing services that satisfy the demands of a variety of companies [17]. Present internet connections have been utilized to enable M2M networks and their services. These types of networks are frequently connected by various protocols [14]. IoT may support companies with solutions that enhance productivity and decision-making in a variety of industries. However, the focus these days is on bringing as many smart devices into our homes and workplaces as possible, making it something that practically everyone can use.

Additionally, IoT services will boost the economy by generating new business opportunities for equipment suppliers, and other participants in the wireless industry. The IoT services, backed by cross-industry cooperation, are expected to benefit billions of people worldwide by improving various economic areas, including transportation, healthcare, and energy [17]. A technology research company anticipates that by 2030, there will be 24.1 billion connected IoT objects worldwide. The report additionally indicates that the largest market for connected devices is expected to be found in the industrial and automobile sectors [16, 18]. In fact, the IoT is still in its earliest stages, but in the end it will have a significant effect on businesses, customers, and society in general [17].

The Internet of things has many practical uses in daily life, ranging from industrial to consumer and business. IoT provides companies with the means to enhance their business strategy and challenges them to reconsider how they run business. On the other hand, customers may benefit from improved connectivity that is caused by the Internet of Things in a number of ways, including increased security and energy efficiency. The usage of IoT technology in an industrial environment is known by several names, including Industry 4.0, the Fourth Industrial Revolution, and the Industrial Internet of Things (IIoT) [16].

Although managing IoT devices can be difficult and demanding, there are certain standard procedures that companies are able to follow to make sure that these devices are safe, reliable, and performing at their best [4]. Through the help of sensors and software,

mobile IoT devices are capable of collecting and analyzing user data and interacting with their users to improve and simplify their lives.

1.3 IOT Architecture

The structure that describes how systems and devices interact and communicate inside the IoT environment is known as IoT architecture. An architecture represents a framework that specifies the physical elements of a network, together with their functional arrangement and configuration [19]. Building and deploying IoT solutions is guided by a framework known as the IoT standard architecture, which describes the essential elements, methods of functioning, and recommended practices for IoT implementations [20]. However, IoT architecture can be seen in a multitude of ways, and each IoT system is unique [19]. The IoT architecture consists of four primary components, and it is designed in layers, as shown in Figure 1.1.

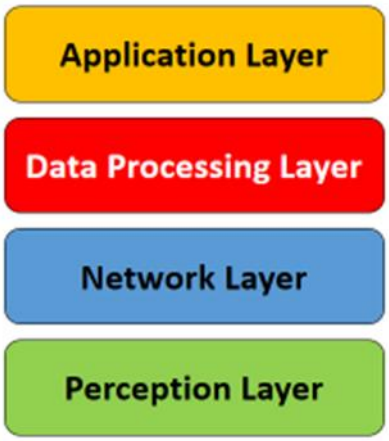


Figure 1. 1: The four-layer architecture for IOT [19].

Regarding the architecture of the IoT, there is not an official worldwide perspective shared by academics and the general public. Researchers have proposed many different kinds of architectures. Some researchers claim that the architecture of the Internet of Things consists of three layers, while other experts advocate for a five-layer architecture. They believe that the three and four-layer architectures can not satisfy application needs because of the IoT security and privacy concerns [21]. The five-layer architecture is an expansion of the IoT's four-layer architecture since it includes an additional layer (Business Layer). The business layer controls the entire IoT system, which includes applications,

revenue, and the security of users. Figure 1.2 depicts IoT layer architectures that include three and five layers.

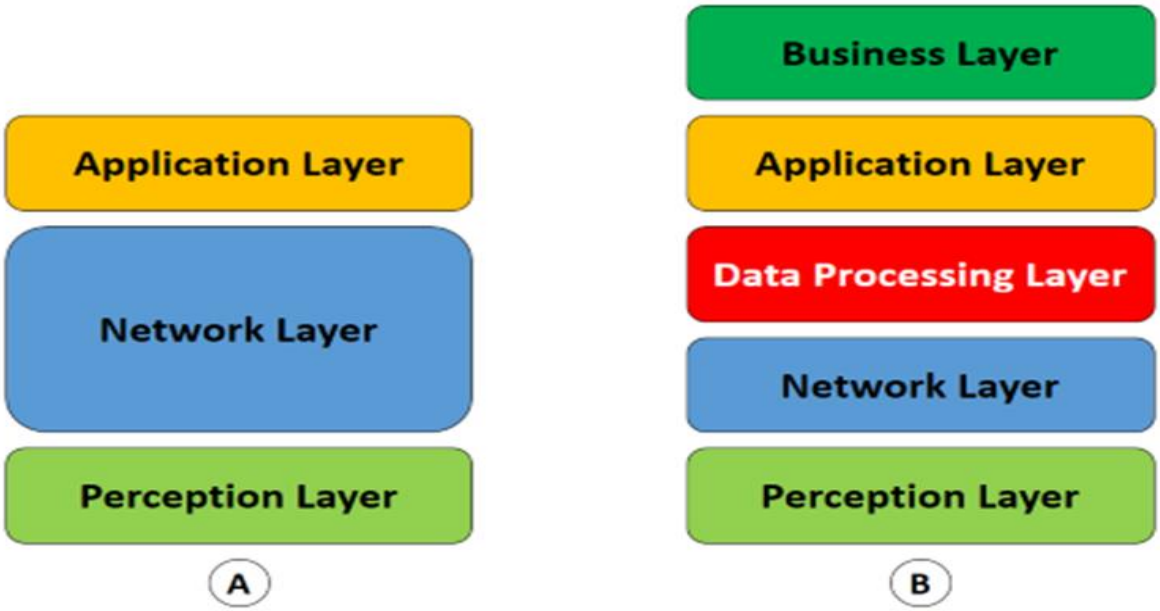


Figure 1. 2: IoT architectures: (A) three layers, (B) five layers [21].

These architectures are composed of many layers and components that work together to smoothly transmit data between devices and applications [22]. The following describes the functioning of the common layers of the IoT architecture.

1.3.1. Perception Layer

The Perception Layer, sometimes referred to as the sensor layer, is the same in the three architectures. It consists of devices, sensors, and actuators that monitor the environment and acquire important data about things like humidity, air quality, temperature, velocity, humidity, pressure, and movement. This layer serves as an interface between the information and physical worlds [4]. Sensors can be utilized to measure physical properties and translate them into signals that IoT devices can handle [14]. The requirements of the sensor determine whether the relationship between environmental factors and the electrical signals is linear or non-linear. Actuators are devices that convert electrical signals into physical actions. They function in the opposite way of sensors, enabling things to take action [23]. Like sensors, the actuator's requirements determine the relationship between the electrical signals and the actuator's function. IoT actuators and sensors are often small, inexpensive, simple, and

energy-saving [4]. The cost and component specifications are crucial factors to take into account while developing an IoT system.

1.3.2. Network Layer

The Network Layer is responsible for transporting data obtained from the gateway node across the entire IoT network. In the context of communication technology, a gateway is a component that connects two networks with dissimilar transmission protocols. In simple terms, a gateway enables connectivity between two protocols by acting as a protocol converter [21]. There are various types of gateways, including VoIP trunk gateways, Email security gateways, media gateways, XML gateways, Cloud storage gateways, and IoT gateways. An IoT gateway is an access point in which a variety of sensors and different components are able to communicate with applications through standardized protocols found in wireless technologies such as WiFi and mobile networks [22].

1.3.3. Data Processing Layer

The processing layer receives data delivered by the network layer. Data preparation and analysis are the primary tasks of the data processing layer. The task given to this layer is to remove unnecessary data and extract the relevant data. Additionally, it solves the big data issue in IoT. Big data refers to the huge quantity of information received, which might have an impact on IoT performance [21]. Through the utilization of artificial intelligence (AI) methods, the processing layer may extract important features from raw data in IoT systems and utilize them for automatic decision making [22]. This layer makes use of a variety of technologies, including big data processing tools, cloud computing, and databases [5].

1.3.4. Application Layer

In an IoT architecture, the application layer provides an essential interface for users to communicate with the system and perform particular functions. It is responsible for providing services to the applications. Services might differ depending on the data that is gathered by sensors, hence they might change for every application [21]. This layer provides control functions, user interfaces, and insightful information by interpreting and using data from IoT devices. Some of its uses include smart cities, manufacturing, healthcare, and smart agriculture [19, 22].

1.4 IoT Gateways

An IoT gateway is a hardware component or software program that connects end devices to the IoT application server [24]. A gateway usually serves as a bridge between M2M devices. However, Traditional gateways do not include any intelligent control features. Due to M2M devices' low processing and memory capacities, these features are critical to IoT systems [25]. Furthermore, incorporating intelligent control features into gateways could make IoT device design easier by offering access to common computing resources [26]. Depending on the application, there are two main methods for building an IoT gateway: the basic gateway and the embedded control gateway.

A basic gateway typically arranges and packetizes information for transmission through the Internet, while an embedded control gateway enhances its capabilities by offering processing power and intelligence to manage local applications. For instance, the embedded control gateway might perform high-level administration functions in addition to analyzing and filtering sensor input. Endpoint complications and costs can be decreased with the help of the intelligent embedded control IoT gateway [26]. The IoT gateway includes a number of features, including: Wireless Connections, RAM, Processor, I/O (GPIO), Operating system (OS), and Security module. Microcontrollers could easily design embedded control gateways by interacting with endpoint devices like actuators and sensors. There are several instances of building IoT gateways with microcontrollers in the published works [27]. Because of its flexibility, compact size, and low energy usage, the microcontroller is a perfect choice for IoT gateway applications. Table 1.1 provides a comparison of the most widely used microcontrollers.

Table 1. 1: A comparison of different microcontrollers.

Name	CPU	CPU Frequency	RAM	Wireless	Cost	Operating Voltage
Raspberry Pi3 Model B	Quad Cortex A53	1.2 GHZ	1 GB	WIFI, Bluetooth	low	5 V
BeagleBone Black	Sitara AM3358-BZCZ100	1 GHz	512 MB	No	low	3.3 V
Arduino Uno	ATMega328	16 MHz	2 KB	No	low	5 V

Table 1.1 (continued)

STM32WBA	Arm Cortex-M33	100 MHz	128 KB	Bluetooth	low	3.6 V
ESP8266	Xtensa L106	80 MHz	160 KB	WIFI	low	3.3 V
ESP32	Xtensa LX6	160 MHz	512 KB	WIFI, Bluetooth	low	3.3 V

In a dynamic environment with high user movement, a mobile gateway could be useful [28]. Since they can connect to a variety of IoT devices using different protocols, mobile devices like tablets and smartphones are able to operate as central communication hubs and serve as IoT gateways. The cost of installing external IoT gateways might be minimized when smartphones and tablets are utilized as gateways. With its diverse range of connectivity features, including Bluetooth, WIFI, and cellular, the smartphone can communicate and work with different IoT devices. The smartphone is thus the user's ultimate tool for controlling and interacting with the IoT systems [29]. A smartphone-based IOT gateway architecture is shown in Figure 1.3.

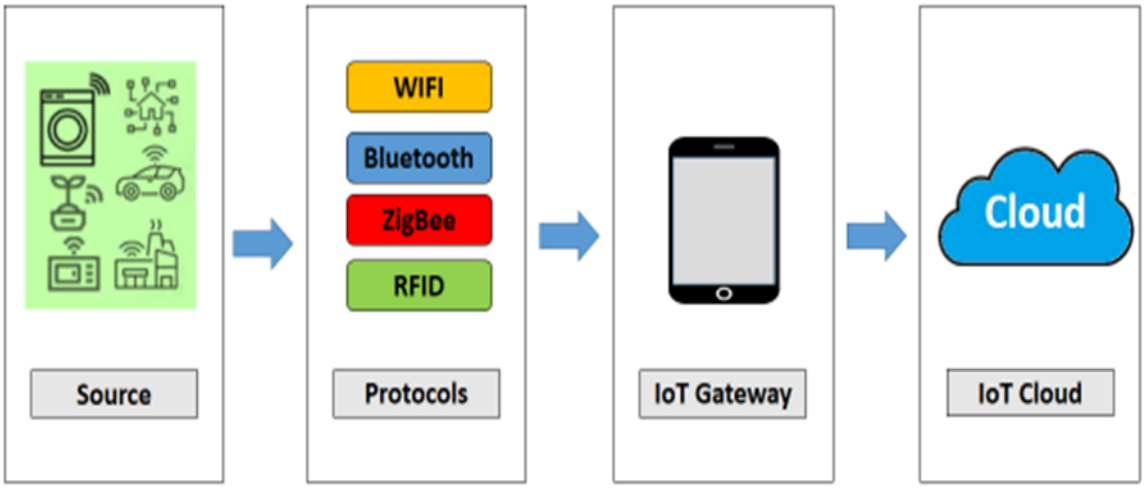


Figure 1. 3: The architecture of an IOT gateway using a smartphone [28].

1.5 IoT Communication Protocols

Connectivity is essential to the Internet of Things' effectiveness and benefits since it facilitates the transfer of data from endpoint devices to central servers via the IoT network. IoT protocols are what make it possible for IoT devices to communicate with each other and share data. Network protocols are utilized for communication between different devices, such as computers and smartphones. However, the standard protocols used by these devices may not be able to match cover distance, latency, and the bandwidth requirements of IoT technology. Despite the simplicity of deployment, IoT devices' communication protocols need to overcome the limitations of current Internet systems in terms of reliability, range, and processing power [30]. Numerous IoT protocols have been developed for various purposes and applications [31]. IoT devices support wireless as well as wired connections for communication [30]. For the three-layer architecture, IoT communication protocols are explained at each layer below, according to the type of connection.

1.5.1. Perception Layer Protocols

Perception Layer protocols facilitate the connection between devices that have limited power and specific requirements in order to deliver services using sensors.

IEEE 802.15: A comprehensive standard protocol known as IEEE 802.15 was created for enabling wireless connectivity in personal area networks, supporting both high-speed multimedia and low-energy IoT devices. It includes a variety of subgroups and standards that address a wide range of wireless communication requirements. Additionally, IEEE 802.15 provides strong network security by encrypting data to guarantee secure connection between devices on the network [30].

Ethernet (IEEE 802.3): The Ethernet network, which was defined by the IEEE in standard 802.3, has been the most successful between local area network (LAN) protocols that utilize wired connections of network devices. Ethernet is a classic protocol used to connect devices in a wired LAN. Thanks to the advancement of technology, data may now be transmitted up to 1000 times faster than when Ethernet first started to appear [32].

LTE: The technology known as LTE, or Long-Term Evolution, is used by cellular service providers to send wireless data to a customer's phone for mobile internet

connectivity. LTE offered flexible bandwidth and frequency, improved effectiveness, peak rate of data transfer, and fast speeds. The speeds that a user may experience at home on a fast cable connection today will be comparable to those of LTE. Due to its unequal modulation and varying data speeds for uplink and downlink, the LTE physical layer is different from others [33].

1.5.2. Network Layer Protocols

The following types of network layer protocols enable communication between devices in an IoT network:

LoRaWAN: LoRaWAN is an LPWAN protocol that is intended for usage with low-power IoT devices. LPWAN (Low Power Wide Area Network) offers a few kilometers of additional range, although most have low data rates [30]. The LoRaWAN protocol is freely available, allowing anyone to establish and run a LoRa network. LoRa is an unlicensed wireless radio frequency technology based on a radio frequency spectrum. Because of its ease of use, energy efficiency, and security, LoRaWAN is widely employed in IoT applications [34].

ZigBee: ZigBee is a solution for low-power, short-range wireless communication, primarily for low-rate sensors in the IoT systems. The ZigBee standard networking protocol is designed specifically for the wireless control field and is focused on wireless control and monitoring. Devices may easily exchange data using very little power across a range of network topologies thanks to the Zigbee protocol. Zigbee enables compatible communication between devices made by different companies [35].

Bluetooth and BLE: Bluetooth is a short-range wireless technology that communicates via radio waves. It allows for communication with multiple devices simultaneously without the need for an interface. It is an accessible protocol that supports multiplier-to-point and point-to-point data transfer. Bluetooth Low Energy (BLE) is an enhanced Bluetooth version that is designed specifically for IoT connectivity. In many IoT applications, BLE is especially desirable since, as its name suggests, it works with less power than traditional Bluetooth [36].

Z-Wave: The Z-Wave protocol is a radio frequency based wireless communication technology that is specifically intended for home application control and state monitoring. This protocol works well for short communication in IoT applications because it is made for small data packets at slow rates [37].

WiFi: Based on the IEEE 802.11 standard, Wi-Fi is a widely used IOT communication technology for wireless local area networks (WLANs). It is constantly being improved to become more responsive, suitable for a variety of devices, and more fast. Security has been improved to satisfy the needs of availability, data privacy, and authentication, while also protecting WiFi connections based on WiFi generation [30].

1.5.3. Application Layer Protocols

Analysis and processing of the information data coming from the lower layers are the responsibilities of the application layer. The following describes four different application layer protocols:

HTTP: The most widely used protocol nowadays is Hypertext Transfer Protocol (HTTP). The primary application of HTTP is as one of the supporting technologies for web browser functionality. HTTP is an application protocol which utilizes the TCP/IP protocol stack, serving as the internet's foundation. The most common use of HTTP in an IoT scenario is to allow devices to POST to a resource on the IoT service which provides the device status [38]. Figure 1.4 displays the HTTP architecture.

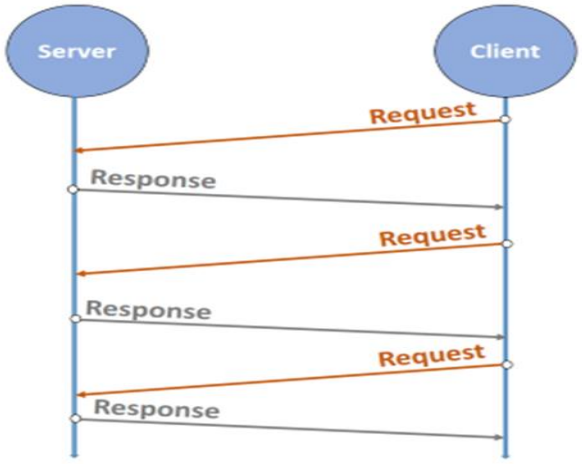


Figure 1. 4: HTTP Architecture [38].

XMPP: Extensible Messaging and Presence Protocol, or XMPP, is an open communication protocol used for real-time applications. XMPP facilitates the decentralized exchange of XML-based communications between clients and servers. It uses a federated framework, allowing people or companies to run their own XMPP server and connect with other users via server-side communication. XMPP utilizes the XML text format and operates on the TCP (Transmission Control Protocol) stack. It works well for closer communications since it includes both request-response and publish-subscribe architectures [39]. Figure 1.5 illustrates the architecture of XMPP.

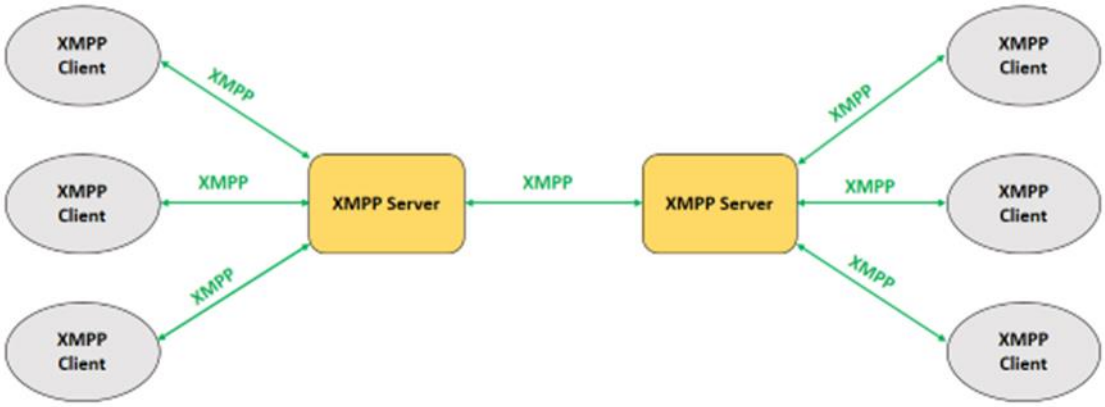


Figure 1. 5: XMPP Architecture [39].

WebSocket: WebSocket is a client-server protocol built on top of TCP. It has less overhead than HTTP and can handle full duplex, two-way communications between the web server and the client. Because of this, WebSocket is effective at transferring little data packets regularly which is particularly helpful for real-time applications. The WebSocket protocol is still quite heavy and overloaded for IoT applications, even though it might be considered an advancement compared to HTTP connection. WebSocket can not automatically provide quality of service guarantees [40]. The architecture of WebSocket is illustrated in Figure 1.6.

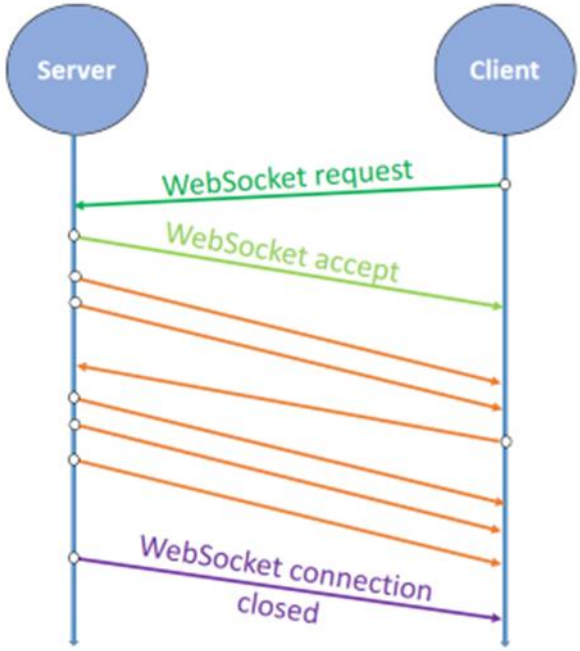


Figure 1. 6: WebSocket Architecture [40].

MQTT: Message Queuing Telemetry Transport, or MQTT, is an exchange messaging protocol that transfers data between clients and servers using the publish-subscribe standard. It is a particular lightweight IoT protocol for environments with limited resources. Via TCP/IP, MQTT is able to operate over network protocols that offer bidirectional, and ordered connections. It is made to communicate effectively between machines (M2M) using a publish-subscribe pattern. When using the publish-subscribe pattern at MQTT, a server known as the broker facilitates the exchange of messages between multiple clients (IoT devices). The messages are distributed to the clients by the broker after being filtered based on the topic and unique identification assigned to each message [41]. MQTT can function and provide sophisticated security services in highly confidential applications and critical systems. As depicted in Figure 1.7 Subscribers are unaware of the source of communications they receive, and publishers are unaware of the destination of messages they send. MQTT uses less electricity, has less overhead, and is faster than HTTP.

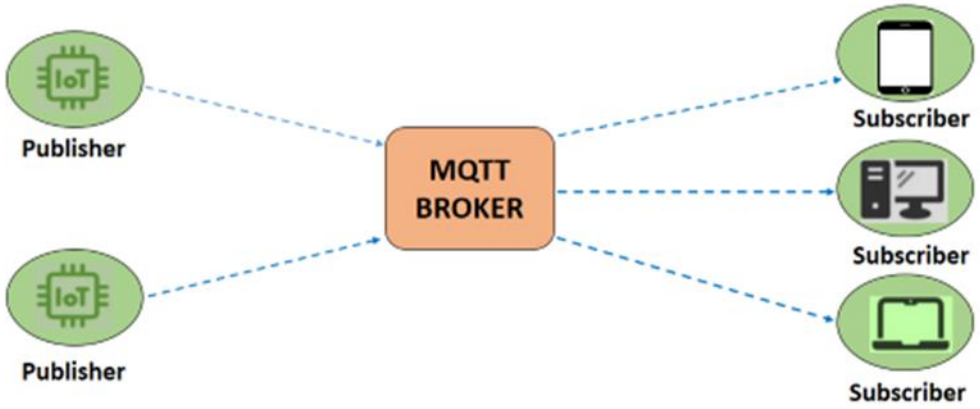


Figure 1. 7: MQTT Architecture [41].

1.6 Comparison of Communication Protocols in IoT

A number of messaging protocols may be chosen, according to the messaging requirements of IoT applications. To decide which protocols are most suited for a given application, it is crucial to know both the advantages and disadvantages of each one. Various parameters are applied to measure the differences among the communication protocols [31]. Comparisons between the IoT communication protocols covered in this chapter are presented in Tables 1.2 and 1.3.

Table 1. 2: A comparison of different application layer protocols.

Protocol	Base Protocol	Standard	Paradigm	Licensing
HTTP	TCP	IETF and W3C	Request/Response	Free
XMPP	TCP	XSF	Request/Response Publish/Subscribe	Open Source
WebSocket	TCP	IETF	Bidirectional	Open Source and commercial
MQTT	TCP	OASIS, Eclipse Foundations	Publish/Subscribe	Open Source

Table 1. 3: A comparison of different network layer protocols.

Protocol	Standard	Data Transfer Rate	Frequency Bands	Range
LoRaWAN	LoRaWAN	50 kbps	415 MHz 868 MHz 915 MHz Depends on geographical location	Up to 15 km
ZigBee	IEEE 802.15.4	250 kbps	2.4 GHz	10-100 m
Bluetooth	IEEE 802.15.1	1-3 Mbps	2.4 GHz	15-30 m
Z-Wave	IEEE 802.15.4	Up to 100 kbps	908.42 MHz (US) Depends on geographical location	30 m
WiFi	IEEE 802.11	Up to 1 Gbps	2.4 GHz 5 GHz	50 to 100 m

1.7 IoT Data Storage and Analytics

1.7.1. Databases in IoT Applications

The data generated from various IoT devices is stored in an updateable, queryable dataset called an IoT database. Massive amounts of time series data are normally generated by IoT devices, which are typically dispersed around both physical and digital spaces. These factors dictate that the optimal database for IoT applications needs to receive and store large amounts of data in real time. Effective real-time data solutions with flexible data modeling, fault tolerance, access to the cloud, and scalability are required to run a real-time IoT application at volume with velocity and minimal latency communications [42].

There are numerous methods for storing captured IoT data. The most popular method is to use non-relational and relational databases. Relational databases include data rows and

columns in tables that are very consistent. Standard database systems were relational databases that used structured query language (SQL) [43]. The most commonly utilized relational databases are:

MySQL: MySQL is the most widely utilized open source relational database management system (DBMS). It is an effective choice for IoT applications due to its many advantages such as availability, open source, flexibility, high performance, scalability, and data security [44]. MySQL is adaptable enough to meet the needs of an IoT technology in any way that is required. It is able to efficiently handle the data processing requirements of the IoT and offers outstanding performance for complex applications. MySQL has demonstrated its ability to manage the massive volumes of data produced by IoT systems. Additionally, MySQL is a good solution when there is demand for many select queries. Numerous well-known websites use MySQL, which is widely recognized as an incredibly safe database technology [45].

PostgreSQL: With fantastic IoT-related capabilities including geospatial support, adaptable data types, query power, and an exciting ecosystem, PostgreSQL is among the most sophisticated databases available. Usually called Postgres, it is a very scalable, extensible database system. PostgreSQL is widely used by businesses worldwide to handle their important information. PostgreSQL is a multiplatform application that runs on Unix, Linux, macOS, and Windows. It supports a wide range of programming languages. Moreover, PostgreSQL adheres to the SQL standard and provides a wide range of extensions that let users improve and personalize the features of their databases. Advanced features available in PostgreSQL include views, triggers, concurrency control, full-text search, and support for JSON [46].

There is now an alternative to relational databases called non-relational databases (NOSQL). NOSQL is not dependent on predefined table configurations or inflexible schemas. There is no need for a special procedure to add columns or records to the collection at any moment. As a result, the number of records in each column does not need to match [44]. The most commonly utilized non-relational databases are:

MongoDB: MongoDB is a scalable and highly performant document-oriented database. In contrast to existing NoSQL databases, its data architecture is built as a document unit,

hence no schema specification is required. Additionally, MongoDB employs a scale-out architecture that allows auto-sharding and is adaptable to hardware evolution. This makes it easier to distribute data automatically among several servers [47]. Data is stored as documents in its binary form known as BSON (Binary JSON) objects, which are JSON-like objects that have been binary encoded. When a document requires an additional field to be added, it may be added without putting the system down, changing the main catalog, or affecting the other documents in the collection. As a result, MongoDB functions better compared to other databases when handling massive volumes of IoT sensor data with regard to resource consumption and long-term storage [48].

Cassandra: Cassandra is a decentralized storage platform which maintains enormous volumes of data on multiple servers. A variety of popular methods are combined by Cassandra to provide availability and scalability. Through indexing, Cassandra's data structure offers fast processing speeds when writing data. Data in Cassandra is indexed with a key, which is a distinct description of the row containing the data. Every row has columns, that carry features, and these columns together constitute a column family. Due to its scalability and capacity to manage real-time data flow, Cassandra is a great option for IoT applications [49].

Every database, whether relational or non-relational, has advantages and disadvantages of its own. So, the most commonly executed query and the application requirements determine which type of database is best for IoT [43].

Data storage is only one component of the Internet of Things environment, which also includes collecting data, transmission, storage, processing, analysis, and applications. Thus, computing systems that are fast, reliable, and effective are becoming more and more necessary. The rapid growth of smart devices and IoT applications has presented novel obstacles for standard cloud computing systems. New methods of real-time data processing and analysis are provided by edge computing and fog computing, which are now recognized as possible solutions to these obstacles [48].

1.7.2. Fog Computing

Fog computing is a network architecture in which data, storage, computation, and applications circulate between the data source and the cloud. The fog node is a crucial part of the architecture utilized in fog computing. There are two types of fog nodes: virtual cloudlets and virtual switches, and physical ones like switches, servers, gateways, and routers. Through processing data on the fog node, fog computing extends the cloud to the edge of the network and makes decentralized computing possible. Every device that may be utilized for computing, storage, and internet access can serve as a fog node in this case [50, 51].

1.7.3. Edge Computing

Edge computing is a relatively recent distributed computing architecture which advocates for data processing and storage near the data source, which was developed in response to the weaknesses of cloud computing's centralized architecture. Devices are necessary for IoT edge computing in order to acquire, analyze, and produce IoT data. In addition, the networking system depends on IoT devices. Edge computing's main benefit is its ability to lower latency, which results in enhanced real-time data processing. This standard solution satisfies the very low latency and bandwidth usage requirements of IoT applications. Edge computing capabilities can be employed to assure fast user response times, network scalability, and big data preprocessing. But in order to fully reap the advantages of edge computing for the IoT, a thorough investigation into processing power and data storage at the network edge is necessary, particularly for IoT applications associated with the next generation [50, 52].

1.7.4. Cloud Computing

Cloud computing is probably the most computationally sophisticated tool currently available. It is a centralized infrastructure technology that provides accessibility to a multitude of resources, such as software services, computers, servers, and databases. The main objective of cloud computing is to offer customers a variety of services in the cloud. There are various cloud deployment options available, like private, public, hybrid, and community. The benefits of cloud computing include cost-effectiveness, streaming services,

scalability, flexibility, and management [52,53]. The growth of IoT has been supported by cloud computing, which provides specialized infrastructure, tools for data analysis, and the ability to store and process enormous amounts of IoT data. Clouds are employed in four different contexts, depending on the kind of capability they provide [54]:

Storage as a Service (StaaS): Cloud applications are permitted to expand outside their restricted servers thanks to StaaS. It enables customers to store their data on distant drives and retrieve it from any location at any time.

Software as a Service (SaaS): Cloud-hosted services have the potential to be beneficial to a wide range of customers. This is a substitute for applications that are executed locally.

Platform as a Service (PaaS): Rather than providing a virtualized infrastructure, cloud systems might offer an extra layer of abstraction by offering the software platform that hosts systems. Everything needed for developers to create, execute, and oversee applications is included in PaaS.

Infrastructure as a Service (IaaS): IaaS is a type of cloud computing which utilizes internet connections to deliver virtualized machine resources. It provides access to a variety of cloud computing resources, such as servers, hardware, network items, and large amounts of storage.

The location, timing, and method of processing and storing data from endpoint devices are the primary differences among edge, fog, and cloud computing. Although less common than cloud, fog and edge computing offer several advantages for enterprises, especially Internet of Things enterprises. The decentralized data storage is customized to specific requirements by these systems, which address numerous problems that IoT cloud computing services are unable to address [51]. Figure 1.8 illustrates the relationship between the edge, fog, and cloud computing concepts.

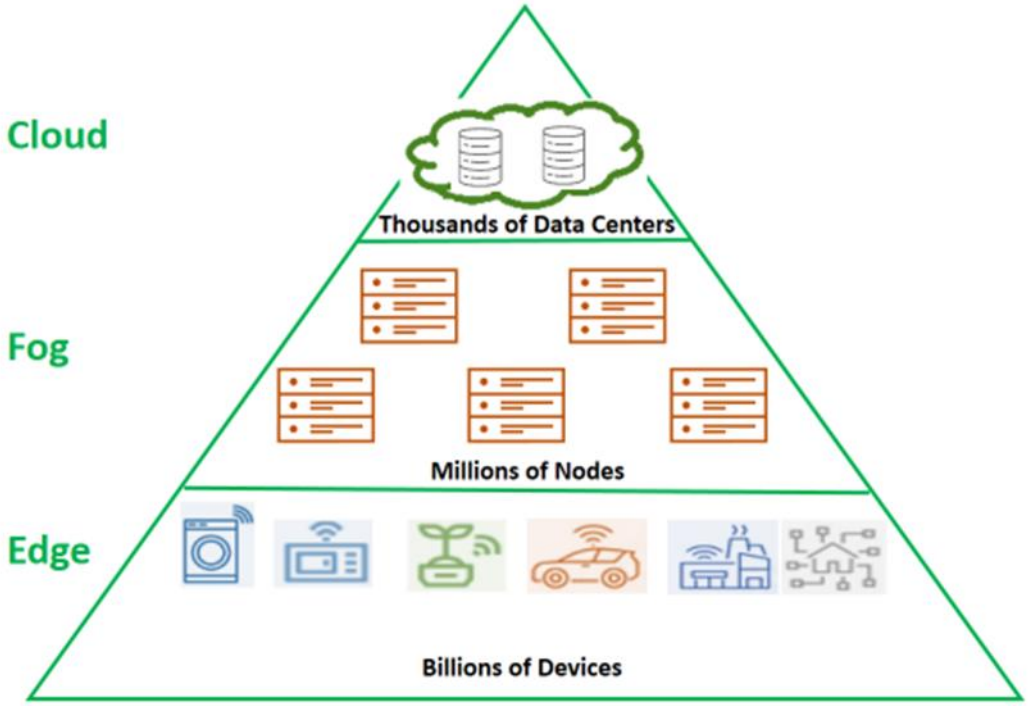


Figure 1. 8: Relation between edge, fog, and cloud computing [50].

1.8 Applications of IoT

Many applications have been touched by the emerging IoT, these applications are categorized according to their size, scope, influence, reliability, customer experience, variation, and availability. There are now a huge number of IoT applications accessible, and there are plenty more in development that should enhance our quality of life. Some of these applications are:

1.8.1. Smart Transportation

A key indicator of how well life is going in today's cities is smart transportation. The IoT improves transportation systems by giving significant real-time data to ensure that machines, managers, and customers may decide on specific actions at the appropriate time. Challenges of the transportation systems involve traffic jams, dangerous driving environments, pollution, and safety. One of the primary challenges facing society is traffic congestion. The traffic condition and the road quality will both be improved by predicting or planning for traffic. An intelligent road surface monitoring system can collect enough data about the

condition of the road from a variety of IoT devices and sensors. This will consequently provide intelligence and reliability to the system [55].

1.8.2. Smart Farming

The farming industry needs technologies such as robotics, automation, drones, artificial intelligence (AI), and IoT in order to satisfy the increasing demands for food. Agricultural IoT (AIoT) has several fundamental elements, including wireless communication technologies, internet access, sensors, detected and transferred data, etc [56].

1.8.3. Healthcare

The primary uses of IoT in healthcare are in situations involving assisted living. Medical equipment management, inventory tracking, and patient monitoring are all possible with IoT devices in healthcare environments like hospitals. In addition to monitoring environmental parameters like air quality, humidity, and temperature, IoT sensors also have the ability to stop the spread of infections. It is possible to put sensors on patients' health monitoring equipment. The data obtained from these sensors can be made accessible to doctors, and family members via web pages in order to enhance therapy [57].

1.8.4. Smart Home

The IoT growth is driving increased interest in home automation research and implementation. Since automated homes and smart connectivity have become inexpensive and easy, smart homes have become more and more popular in recent years. Smart homes provide increased elegance, safety, conserving energy, and comfort [55].

1.8.5. Manufacturing

Manufacturing experiences a tremendous technological change driven by the integration of IoT. The IoT is being used by all industries to automate operations, utilize edge processing, and obtain useful data across the internet. Due to its ability to monitor and optimize the manufacturing process through a network of interconnected devices, sensors, and software, the IoT is significantly contributing to the transformation of traditional manufacturers into intelligent manufacturing in Industry 4.0. In fact, Industrial Internet of

Things (IIoT) technologies open up possibilities for Industry 4.0. In Industry 4.0, IoT sensors allow machines to exchange data, coordinate actions, and interact with each other. IoT devices increase robotic productivity and efficiency by allowing machines to share data, which also increases safety and lowers the need for unexpected repairs [58, 59].

1.9 Artificial Intelligence for IoT Data Analytics

The term artificial intelligence (AI) describes a machine's capacity for learning and making decisions without human involvement. With the help of data and intelligence embedded in devices connected to the network, IoT technologies interact with environment to assist people in going about their daily lives in a simple and natural way. AI has the potential to enhance efficiency and simplify operations in the IoT by introducing human-like decision-making and intelligence [60].

AI techniques include machine learning (ML) and deep learning (DL). ML can operate in dynamic networks without the need for humans or complex mathematical equations. To identify and react to human action, ML approaches such as supervised, unsupervised, and reinforcement learning can be utilized [11].

1.9.1. Machine Learning (ML)

In ML, supervised learning is the most popular approach. It involves utilizing a trained dataset, to classify the output depending on the input. Regression and classification learning are two categories of supervised learning. The following examples present various types of classification learning.

Decision Tree (DT): The DT is a typical supervised learning technique, similar to a tree with leaves and branches. It is a hierarchical model which includes leaf nodes, internal nodes, branches, and a root node. Every node in the leaf indicates the final prediction, and every branch relates to a feature value. Compared to other ML techniques, DT offers the advantages of being transparent, having an easy-to-implement design, and handling big data samples [11].

Random Forest (RF): Known for its ability to address regression and classification problems in ML, RF is a common ML algorithm classified under the supervised learning

technique. The performance of many DT algorithms is combined by the RF algorithm for classification. This technique is a useful tool for a variety of ML predictive tasks because of its strength in handling complicated datasets and mitigating overfitting [61].

K-nearest Neighbor (KNN): The KNN algorithm is a supervised ML approach developed for dealing with regression and classification problems. It is predicated on the concept that comparable data points typically have comparable labels or values. To make predictions, KNN uses a selected distance metric, like Euclidean distance, to determine the distance between each instance of training and the input data point [11].

Support Vector Machines (SVM): SVM is a sophisticated ML technique that may be utilized for regression, linear or nonlinear classification, and identifying outliers. The SVM algorithm's primary goal is to identify the best hyperplane in a space with N dimensions which may be used to divide data points into various feature space categories [61].

Naive Bayes (NB): The NB is a probabilistic ML algorithm that utilizes Bayes' theorem. To be able to use Bayesian probability and predict the probable results, this type of supervised learning technique requires prior data. Based on a given set of evidence (E), Bayes Theory determines a hypothesis (H). It is related to two factors: the probability $P(H)$ of the hypothesis prior to the evidence and the probability $P(H|E)$ following the evidence [11]. The following formula explains the Bayes Theory:

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)} \qquad (1.1)$$

- $P(H|E)$ indicates how event H transpires in the context of event E.
- $P(E|H)$ is the rate of event E when event H occurs first.
- $P(H)$ represents the previous probability of class.
- $P(E)$ indicates the predictor's previous probability.

K-mean Clustering: K-means clustering is an unsupervised learning method that uses unlabeled data, as opposed to supervised learning. It is a popular cluster analysis technique which aims to divide a set of items into K clusters in order to minimize the sum of the squared distances among each item and the chosen cluster mean. This method divides the provided data samples into smaller groups so that they can be categorized as a cluster. The centroid

of the cluster is first randomly assigned to the space. Next, data points are assigned to one of the K clusters based on their proximity to the cluster center. Following the assignment of each point to a cluster, new cluster centroids are created. Repeatedly, this procedure continues until it finds an acceptable cluster [62].

Reinforcement Learning (RL): Through RL, a machine can learn from interacting with the environment by taking behaviors that optimize the overall feedback, just like humans do. Feedback may take the form of an award that is dependent on completing the assigned task. The system learns by trial and error in reinforcement learning, where no predefined behaviors are assigned to any specific task. The agent can find and use the most effective strategy based on its expertise to obtain the most reward via trial and error [11].

Deep Neural Networks (DNNs): DNN is widely employed in the area of automated classification due to their precision and flexibility. In general, a DNN is an ML technique built on an artificial neural network (ANN), which imitates the architecture and workings of a human neural network. A DNN is composed of mathematically related layers, such as edges and nodes. It offers a major benefit over typical ML algorithms in that they can extract features at many levels of abstraction, which allows them to learn deeper patterns. Three layers make up an ANN: input, hidden, and output. The system is referred to be a DNN if the total number of hidden layers is three or more [11, 63].

1.9.2. Deep Learning (DL)

The DL, which is a branch of ML, includes methods and computer models that imitate the structure of the biological neural networks found in the brain. The brain attempts to make sense of new information by comparing it to previously learned knowledge. DL uses the same method that the brain uses to interpret information by labeling and classifying the objects [12]. The most often used techniques are listed below:

Convolutional Neural Networks (CNNs): The most widely used DL is CNN due to its ease of training and robustness. There are various structural elements in the CNN architecture, including convolution layers, pooling layers, and fully connected layers. A basic architecture comprises multiple repeats of a collection of convolution layers and a pooling layer, then several fully connected layers. It works by using a series of basic

convolution and deconvolution operators at various scales. Convolution is the process of applying several filters to aggregate data from pixels that have been clustered together. The filters vary from the current layer to the ones that follow, and their function provides the input for the following layer [64].

Recurrent Neural Networks (RNNs): An RNN is a form of ANN that processes sequential or time series data. It derives its name from the fact that it executes a single task for each element in a sequence, with the final result depending on prior calculations. A different means to conceptualize RNNs is as having an internal memory that stores data about previous calculations. RNN works on the basic principle of creating a prediction based on both the input data and the previous outputs. The idea makes a lot of sense to create neural networks that can advance values over time [65].

Long Short-Term Memory (LSTM): LSTM networks are a particular kind of RNN that may learn order dependence in predicted sequence tasks. LSTMs are made to solve the vanishing gradient issue that RNNs encounter. Since LSTMs use specialized memory cells including input, forget, and output gates, they are capable of capturing long-term dependencies in time series data. LSTMs are especially useful for forecasting tasks because of their capacity to retain relevant data over extended periods of time [65]. Figure 1.9 provides an illustration of the LSTM.

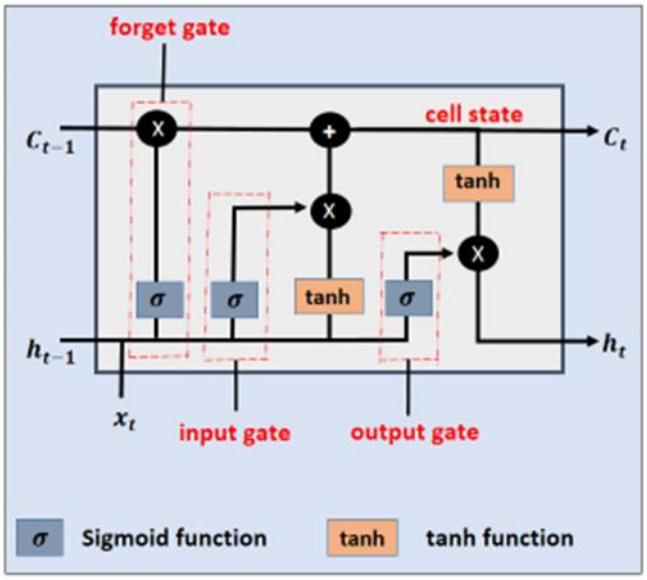


Figure 1. 9: The illustrations of LSTM model [65].

Gated Recurrent Unit (GRU): GRU is defined as a variant of the LSTM since they are designed similarly and under certain situations, produce equally superior results. The key distinction among GRU and LSTM is how they handle memory cell state. A candidate activation vector is used in GRU to replace the memory cell state, and it is changed via the update and reset gates. When updating the hidden state, the update gate decides the amount of the new input that can be utilized, and the reset gate decides how much of the prior hidden state should be forgotten. Based on the new hidden state, the GRU calculates its output [66]. The GRU illustration is shown in Figure 1.10.

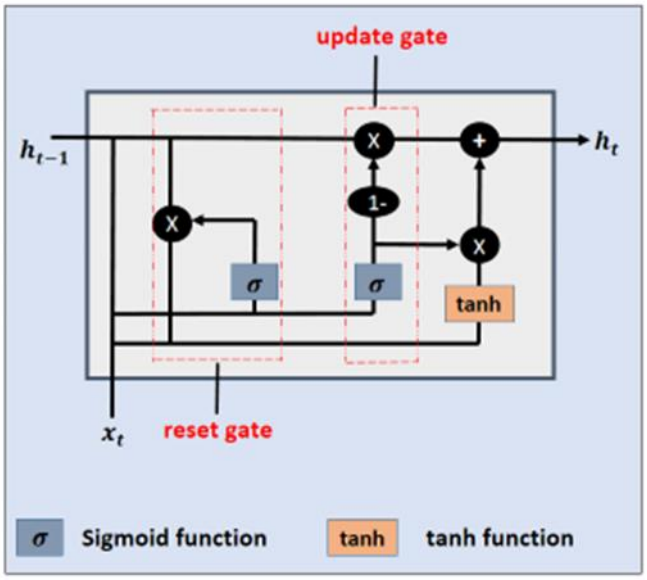


Figure 1. 10: The illustrations of GRU model [66].

1.9.3. Ensemble Learning (EL)

The main idea behind EL is to train a number of base models to be ensemble participants, then combine their predictions to produce one final result which should perform more effectively than all other individual models [67]. Bias and variance are two fundamental ideas in EL, which allow learners to acquire information with poor accuracy and enhance accuracy. A bad balance between variance and bias could result from insufficient accuracy improvements. The bias can be defined as the variation among the predicted and actual values. In contrast, variance refers to the amount of variation between the predicted values [68]. The following is a list of the most popular EL methods:

Averaging: Averaging involves making numerous predictions for every data point. This approach uses an average of predictions from all models to generate the final prediction.

Weighted Average: This is an extension of the averaging approach. The Weighted Average approach assigns various weights to each model, indicating the importance of each individual model for prediction.

Max Voting: The max voting approach is commonly employed for classification issues. Several models are employed in this approach to generate predictions for every data point. A vote is taken based on every model's prediction. The final prediction is obtained using the predictions generated by most of the models.

Boosting: Boosting is a sequential procedure in which each subsequent model attempts to rectify or correct the mistakes of the previous model. Every subsequent model is dependent on the model that came before it. Boosting strategies combine numerous weak learners to create a powerful learner. Though they are performing effectively for a portion of the dataset, the individual models may not improve the accuracy of the whole dataset. Hence, every individual model significantly boosts (enhances) the ensemble's performance. CatBoost, Light GBM, XGBM, GBM, and AdaBoost are some popular boosting algorithms.

Bagging: Bagging, also known as bootstrap aggregation, is the process of integrating the results of many models to produce a more generalized result. It uses bootstrapping-sampling methods to generate several bags (subsets) from the training data set via replacement. An independent base model is trained with every subset. Since the subsets are varied, the models are able to learn unique patterns from the data.

Stacking: Stacking is an EL approach that combines predictions from multiple models to create a new model, which is then used to produce predictions using the testing data set. It aims to improve a classifier's capacity for prediction.

Blending: Blending is similar to stacking, except it only makes predictions based on the holdout (validation) set from the training dataset. In other words, predictions are made exclusively on the validation dataset, not the training dataset. The final model is built using the validation dataset and the predictions, and it is then applied to the test dataset.

1.10 Big Data

Big data refers to extraordinarily large and complex data collections that may be processed faster than traditional data storage and analysis systems. It is just a term used to describe larger, more complex data collections, especially those that come from new data sources. These data volumes are so large that traditional data processing methods cannot manage them. The IoT is widely recognized as a primary source of big data due to its ability to connect an extensive amount of intelligent devices to the Internet and provide information about the condition of their environments on a regular basis. The main benefit of big data analytics is its ability to identify and extract significant patterns from massive amounts of raw data, which leads to deeper kinds of knowledge for trend prediction and making decisions [6, 69].

Big data tools refer to the technologies or methods used for processing data that can be categorized as big data in an efficient manner. Numerous academic papers have discussed the broad characteristics of big data from various angles including volume, velocity, and variety. Big data is typically described by six characteristics, called the 6V's [70]. The 6Vs, or the fundamental characteristics of big data, are depicted in Figure 1.11.

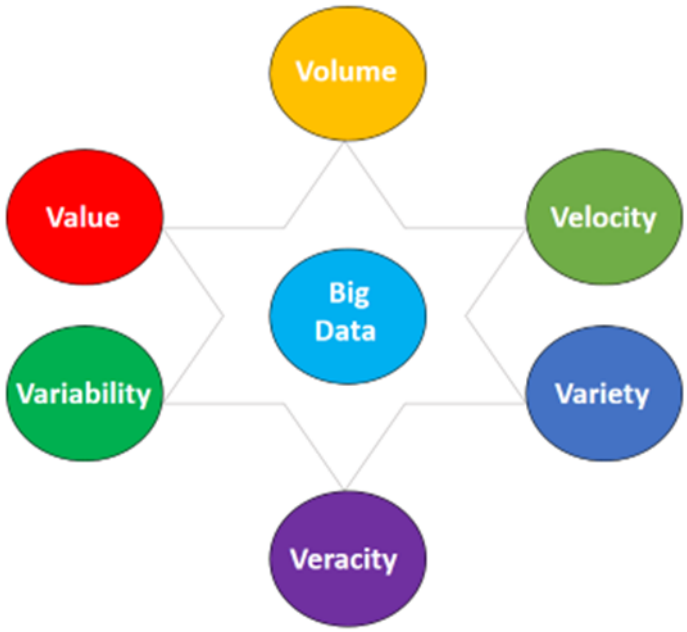


Figure 1. 11: The 6V’s of big data [70].

Given that they help us comprehend the nature of big data, these six fundamental components are crucial starting points when using big data. The 6 V's are defined as follows:

Volume: Volume is an indicator of the data that is available. It is similar to the foundation of big data since it represents the original size and quantity of data acquired. Big data can be defined as data that has a sufficiently high volume. Despite this, the definition of big data is inflexible and prone to change in response to changes in the market for processing power.

Velocity: The speed at which IoT big data is produced and processed is sufficient to support real time availability. This is a crucial component for companies which require their data to move rapidly and be accessible when needed to enable optimal business decisions.

Variety: Variety describes the wide range of data kinds. IoT can generate a range of data kinds, including unstructured, semi-structured, and structured data.

Veracity: Veracity relates to the data's quality, consistency, and reliability, which results in precise analytics. It is especially important for IoT systems that use crowd-sensing data to maintain this characteristic.

Variability: Variability describes differences in the use or flow of big data. Different components that generate data may have irregular data flows, based on the type of IoT application. Further, a data source may have varying data load rates depending on particular periods.

Value: Value is the converting of big data into useful insights and data that offer companies a competitive advantage. Big data gets much more valuable based on the information that can be obtained from it, therefore being able to extract value from it is essential.

Massive amounts of data processing are currently possible and supported with the help of specialized Big Data frameworks. They facilitate the rapid analysis and organization of large amounts of real time data. Apache Hadoop, Flink, Storm, and Spark are some examples of big data frameworks [70].

Apache Hadoop: Apache Hadoop is a batch processing platform which offers fault tolerance and fault scalability. Hadoop allows programs to run on numerous nodes and manages petabytes of data. Additionally, before being transferred to the Hadoop cluster's nodes, the log data is divided into chunks [13].

Apache Flink: Flink is an open source framework that can process data in both batch and real time modes. It has various advantages, including fault tolerance and massive scale processing [71].

Apache Storm: Storm is an open source platform that processes enormous amounts of structured and unstructured data. It is a fault tolerant framework designed for ML, iterative and sequential calculation, and real time data processing. A million tuples can be processed by a node of Storm every second [71].

Apache Spark: The Apache Spark framework will be described in detail in Chapter 3.

1.11 IoT Security

IoT security is a field of research that focuses on protecting IoT networks and connected devices. IoT devices need a special set of cybersecurity requirements due to the ways in which they differ from traditional mobile devices in terms of functionality. IoT devices have become widely available to users since they can be utilized for a variety of tasks across a public network. IoT advances human life by making it easier, but it also increases the risk to users' privacy by posing various dangers and attacks. Since some IoT devices can be accessed by everyone, anywhere, without the owner's consent, the security of IoT devices has emerged as a critical concern. The IoT devices need to be protected by a variety of security systems [11, 70, 72]. While building IoT security solutions, a number of features have to be taken into account. The following aspects should be addressed when building security protocols for preventing attacks against IoT systems.

Identification: Every entity in an IoT system has to be able to recognize other participants and be informed about other entities inside the network. Additionally, entities should be able to differentiate between benign and possibly malicious entities. The IoT system needs an efficient identifying mechanism that can offer robust security with system limitations [72].

Authorization: Authorization refers to the user's accessibility within an IoT system, allowing access to authorized clients who can enter, track, and utilize data contained in the network. The clients with authorization on the system may also give instructions to it [11].

Availability: Availability is the likelihood that a system will function at any moment in time. The system has to be accessible when needed, as well as its services. Data availability is crucial for surveillance and safety applications. Lack of knowledge can cause problems with home and workplace automation as well as economic damage in commercial applications [13].

Integrity: The integrity attribute guarantees that, when utilizing a wireless network for communication, only users with permission can change the data on IoT devices. The long-term functionality of IoT devices will be impacted if this feature is harmed in any way due to unusual examination during data storage [11].

Privacy: Ensuring that the rights users have over their utilization of personal information are properly respected is vital while managing, analyzing, storing, and deleting data. Since attackers may detect an IoT device's physical position and decode the data, it is challenging to protect most data from unknown users [72].

Intrusion detection systems (IDS) have been associated with the main problems of IoT security. Generally, IDS are used to identify anomalies in the network. An anomaly is a malfunction in the IoT device network that can be produced by an attack, intrusion, or just a simple flaw. In AI based anomaly detection, significant factors influencing IDS reliability include IDS methodologies, data preprocessing, feature extraction, and appropriate feature selection [11].

1.12 IoT Search Engine

Large volumes of data are produced by IoT systems because they share more data amongst smart devices than they do between humans and devices. In comparison to old static data, this data is updated frequently and may dynamically characterize the state of the associated systems to give a more accurate and precise representation of a system's condition. Hence, there is a pressing need for an IoT search engine that offers query resolution services to help consumers locate pertinent IoT data rapidly, similar to the way

online search engines do [73, 74]. The management and search problems associated with IoT systems are addressed by Internet of Things search engines (IoTSE). IoTSE is designed to function similarly to a standard web browser, except that all of the IoT data could be transformed into URL links so that users may access and examine the data in response to their requests. IoTSE can perform data collection, indexing, and organization functions, similar to web search engines [75]. Figure 1.12 illustrates the IoTSE's framework in detail.

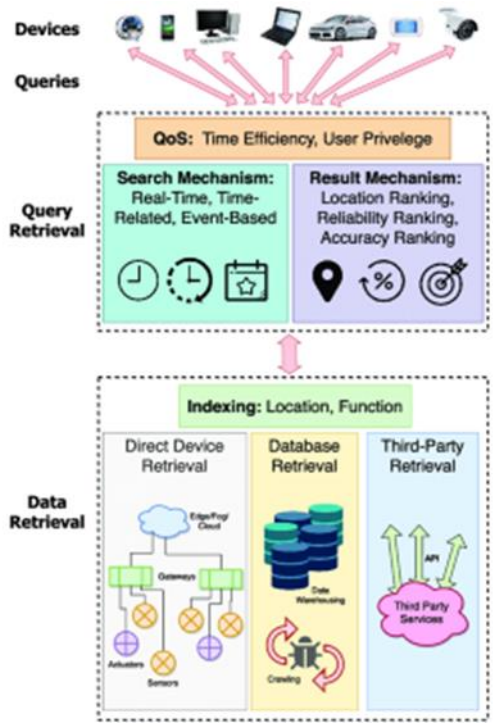


Figure 1. 12: An overall framework for the IoTSE [76].

An IoTSE is intended to look through a variety of IoT resources, such as devices and data. IoTSE functions similarly to web search engines in that they provide the necessary IoT data, IoT resources, or combinations of both in response to user queries [74]. The key elements of the IoTSE are shown below.

IoT Resources: IoT transforms physical things with embedded technology (computing and networking elements) into digitally flexible smart resources, including networking devices, storage, actuators, and smart sensors. The IoT search procedure abstracts each IoT device as an IoT resource [74].

IoT Data: The IoTSE's fundamental component is the IoT data, which serves as its primary search target. Typically, gathered data is used to build a digital model that depicts the state or past of the associated environment [74].

Search Space: A search space is required for IoT search, just like it is for web search engines. The search space consists of a collection of IoT resources with clearly defined data structures. Because IoT resources are dynamic and update significantly faster than web content, the search space available for IoT search is far larger than for standard web search [74].

IoT Query: Both humans and IoT devices can submit queries in an IoT search. Data interchange and communication between IoT devices from different planes are required. However, to accomplish intelligence and automation, IoT devices should specifically send queries [74].

Edge Computing Nodes: Computational activities from cloud infrastructures may be transferred to edge and fog nodes located near IoT devices through distributed computing. The search functions can be carried out by the edge computing nodes, which can be easily expanded. In addition, edge nodes are employed in the IoTSE to harvest IoT data and refresh the search space index [74].

Middleware: As intermediary between applications and IoT resources, IoT middleware is another crucial part of the IoTSE. middleware offers a basic query operating platform while hiding diverse IoT resources. Furthermore, because the middleware acts as an IoT service provider, it may be placed anywhere there is sufficient processing capacity to perform the required calculations, such as edge computing nodes, cloud servers, and IoT gateways [74].

1.13 Conclusion

There will probably be a lot more things connected in the near future due to the rapid growth of the Internet of things (IoT). It is very likely that the world will become extremely connected in the future. The architecture of the IoT system and its supporting network were covered in this chapter. We demonstrated a few IoT application areas. Various methods and protocols have been discussed for the aim of connecting IoT devices. Additionally, we gave

a brief description of the machine learning and deep learning approaches that are utilized in IoT data processing. Then an overview of big data, IoT security, and IoT search engines was presented.

CHAPTER 2: IOT BASED ROAD SURFACE CONDITION (RSC) MONITORING USING HYBRID DEEP LEARNING MODELS

2.1 Introduction

Intelligent transportation systems (ITS) offer creative applications concerning traffic management and many forms of transportation. This enables road users to utilize transportation networks in a safer, more effective, and intelligent manner. The application of Internet of Things (IoT) in the transportation sector has gained popularity recently as a consequence of present challenges and technological advancements. The use of IoT in transportation applications is being driven by technological advancements, which make cities smarter and their systems easier to manage. In addition, IoT in transportation refers to a vast network of embedded smart devices, like actuators and sensors. These sensors capture data about the external physical environment, which is then transmitted to specialist software to provide meaningful information, thus improving infrastructure design and maintenance as well as safety [77, 78].

In recent years, monitoring of road anomalies has grown to become a major topic in driving safety studies. Every day, around 3,700 people lose their lives in road accidents, for a total of 1.35 million deaths worldwide each year. Actually, the number of vehicles on the road is increasing, which has led to additional damage to the road surface. This might result in significant vehicle damage and an increase in traffic accidents. Many people have complaints about the state of the roads in their hometowns or places of work due to damages caused by poor road surfaces. However, maintaining top-quality road infrastructure is expensive since it has to be continuously monitored and repaired [79, 80].

Detecting anomalies in the road surface is crucial for road maintenance, improving automated driving, and reducing the rate of accidents. Here, road surface anomalies refer to speed bumps, speed humps, manholes, potholes, cracks, etc. In this context, researchers have been trying to detect road surface anomalies by creating various methodologies and building road surface condition (RSC) monitoring systems.

The term artificial intelligence (AI) is used to describe a variety of algorithms and concept solutions that may be used to accurately identify anomalies on the road. These AI methods include machine learning (ML) and deep learning (DL), which are used to train

classifiers that discover correlations and patterns between the input features and corresponding road types [78, 81, 82].

This chapter will cover some of the most common approaches for road data analysis and RSC monitoring. In addition, we will propose architectures for RSC monitoring and frameworks for detecting road anomalies.

2.2 Background

For the related work, we start with providing an overview of road surface condition (RSC) monitoring approaches. Afterward, we present techniques for detecting road anomalies using the vibration-based approach. Finally, we highlight some of the most important RSC monitoring research contributions.

2.2.1. Road Surface Condition Monitoring Approaches

Several studies have been carried out in the literature to monitor road surface conditions (RSC). These studies can be divided into three major approaches: vision-based, laser-scanning-based, and vibration-based.

A vision-based approach primarily employs vehicle-mounted cameras to capture 2D images or videos of the road surface. This approach detects road defects using image processing analysis. However, a number of external factors such as lighting and the influence of shadows, have a negative impact on the vision-based approach [83].

With a laser-scanning-based approach, the road conditions are assessed using specialized cars equipped with laser sensors. This approach employs 3D laser scanning to generate precise road surface models. To identify road surface anomalies, a base model is compared with these 3D digital models. Nevertheless, when it comes to monitoring vast road networks, the laser-scanning-based method is very expensive and time consuming [84].

A vibration-based approach allows for the detection of road anomalies through analyzing a moving vehicle's vibration rate acquired by motion sensors. Unlike the first two methods, which are both expensive, the vibration-based approach is widely available and low-cost [85].

2.2.2. Road Anomalies Classification Approaches

The studies of road surface condition (RSC) monitoring focus on the identification of anomalies in the road surface through time series information collected with sensors. As shown in Figure 2.1, road surface anomalies can be classified as real or man-made. The vibration-based approach detects road anomalies in three stages. Firstly, the condition of the road surface data have to be collected using wearable sensors, which are popular IoT devices and they are easily able to be embedded in modern vehicles. Secondly, the collected data are then preprocessed to smooth the sensor data and remove the noise. Preprocessing also aims to organize data by transforming sensor signals into a format that is more appropriate for the input data. Finally, road surface anomalies are detected by classifying the preprocessed sensor signals based on the pattern of signals. Typically, thresholding, dynamic time warping (DTW), machine learning (ML), and deep learning (DL) techniques have been utilized to identify any unusual variations in sensor values [86, 87].

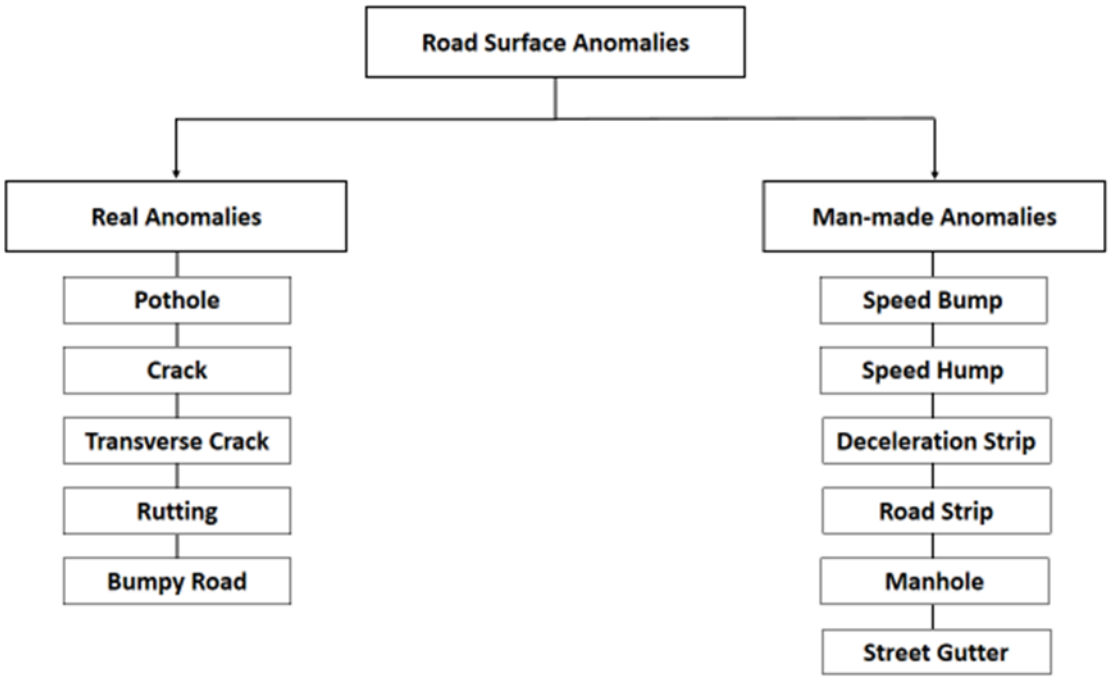


Figure 2. 1: Types of anomalies on the road surface.

2.2.3. A Review of RSC Studies

Several research studies on road surface condition (RSC) monitoring have been published in the literature. Some of the recent RSC studies that have been published are summarized in Table 2.1.

Table 2. 1: A summary of related work in RSC monitoring.

Reference	Approach	Contributions	Objective
S. Sattar et al. [88]	Vibration-based	-Smartphone-based data collection. -Unsupervised ML algorithm (GMM) for road surface anomaly detection.	Building a near-real-time method for classifying road surface anomalies into two categories according to the degree of anomaly severity.
Allouch et al. [89]	Vibration-based	-Automatic collecting and labeling of gyroscope and accelerometer data. - A low-pass filter to eliminate sensor noise and FFT for feature extraction - ML algorithm for road segment classification.	Developing a real-time Android application that plots the road surface anomaly location on a map to automatically determine the quality of the road.
Zhou et al. [90]	Vision-based and Vibration-based	-Smartphones were employed to collect road image and motion sensor data. - A DL model for detecting manhole covers from a road images and ML model for classifying the detected manhole covers.	Detecting and classifying road manhole cover subsidence into three levels: poor, average, and good.

Table 2.1 (continued)

Yang et al. [83]	Vision-based	-Collecting road image via a digital camera that has a polarization filter attached. - A wavelet packet transform for feature extraction and ML algorithm (SVM) for road surface anomaly detection.	Determining if the road surface is wet, dry, snowy, or ice-covered.
Higashimoto et al. [84]	Laser-scanning-based	-Extraction of road surface environments using LiDAR point cloud data. -A ray ground filter to separate point cloud data into ground and non-ground. -Utilizing the point cloud data's standard deviation of the reflection intensity to identify the road surface.	Developing a real time road surface environment detection system based on LiDAR to enable self-driving cars to teleoperate in undeveloped areas.
Varona et al. [85]	Vibration-based	-Data collecting via smartphones. -Classifying road segments using DL algorithms. -Data augmentation approaches for maintaining the dataset balanced.	Automatically differentiating between a vehicle's instabilities generated by potholes and that generated on by speed bumps or driving behavior.

Table 2.1 (continued)

Setiawan et al. [91]	Vibration-based	-Data collecting through smartphone and data labeling via video camera. - Classifying road segments using DL model (DCCN).	Proposing an approach to balance the training of an autonomous road surface evaluation system by augmenting smartphone sensor data using an unrolled GAN.
Du et al. [92]	Vibration-based	- A Gaussian background model for feature extraction and ML algorithm (kNN) for classifying abnormal pavement types.	Employing a smartphone's acceleration sensor to easily and cheaply identify abnormal road surfaces
Wu et al. [93]	Vibration-based	-Feature extraction using wavelet decomposition, frequency domain, and temporal domain - ML algorithm for road pothole detection.	Utilizing vibration data from smartphones to develop an automatic pothole detecting system.
Baldini et al. [81]	Vibration-based	-Data collecting via inertial measurement unit (IMU). -Time-frequency features are put into the CNN.	Detecting and identifying anomalies in the road surface by combining CNN and time-frequency transform.

2.3 Proposed Methodology for Road Anomalies Detection

The proposed frameworks for detecting road surface anomalies aim to distinguish three types of road surfaces: Real anomalies (potholes and cracks), man-made anomalies (speed

bumps and speed humps), and smooth roads. This methodology generally consists mainly of three phases: (1) data collection, (2) data preprocessing, and (3) classification.

As opposed to previous vibration-based approach studies, which primarily employ cameras for sensor data labeling and machine learning or deep learning for data classification. Furthermore, they rarely distinguished between man-made road anomalies (which cannot be considered as road defects) and real road anomalies. The novelty of our work comes from the use of hybrid deep learning models to improve classification performance in three types of road surfaces, namely smooth road, real road anomalies, and man-made road anomalies. In addition, we proposed novel labeling techniques that allow us to label sensor data in real time utilizing a smartphone application. These new techniques of data labeling are more effective than using images from cameras since they are not impacted by light conditions or weather.

2.3.1. Data Collection

In general, a vehicle vibrates more while traveling over road surface anomalies, like speed bumps, speed humps, potholes, or cracks, than when moving over flat road surfaces. Therefore, the vibration-based method is an excellent option for collecting road data. With the goal of collecting road data using a low-cost technology that is unaffected by light conditions or weather, we utilized motion sensors embedded in IoT devices or smartphones. Motion sensors like accelerometers, gyroscopes, and magnetometers are excellent choices for obtaining vibration data from vehicles since they are capable of measuring vibration, rotation, and movement of devices. The Global Positioning System (GPS) is the best option for determining the location of road anomalies since it offers current location information such as latitude, longitude, altitude, and speed. To establish the ground truth, an adequate data sampling rate should be chosen to capture all road surface information [86].

Here, we employed the three axes (X, Y, and Z) of gyroscope and accelerometer sensors, as well as orientation angles, namely Azimuth, Pitch, and Roll which are computed by using an accelerometer in conjunction with a magnetometer. The HTTP protocol is used to send the sensor data to the MySQL dataset for IoT data storage. Table 2.2 contains a description of the sensors utilized. However, these sensors are sensitive to

the location of the device in the vehicle. Usually the device is attached near the gearbox, near the gearshift, dashboard, or windshield. In the data collection phase, data labeling was done in real time, with two people in the car, one driving and the other labeling. Road data have been collected as a time series, due to the timestamp included in each raw sensor data [85, 88].

Table 2. 2: A description of the sensors utilized to detect road surface anomalies.

Sensor Name	Units of measure	Description
Accelerometer	m/s^2	Acceleration force over the X-axis
		Acceleration force over the Y-axis
		Acceleration force over the Z-axis
Gyroscope	rad/s	Rotation rate around the X-axis
		Rotation rate around the Y-axis
		Rotation rate around the Z-axis
Magnetometer	μT	Strength of the geomagnetic field through the X-axis
		Strength of the geomagnetic field through the Y-axis
		Strength of the geomagnetic field through the Z-axis
Accelerometer and Magnetometer (orientation angles)	Degrees	Pitch (angle around the X-axis)
		Roll (angle around the Y-axis)
		Azimuth (angle around the Z-axis)
GPS	Degrees	Current latitude value
		Current longitude value
		Current altitude value
	m/s	Current Speed value

2.3.2. Data Preprocessing

Several data preprocessing techniques were done to transform the acquired sensor readings into clean and organized data, which could enhance the success rate of road surface anomalies detection. The preprocessing techniques that utilized for transforming road data are presented in the following.

2.3.2.1 Data Filtering

Longitudinal vibration peaks are considered by road engineers to be road surface anomalies. They all produce vibrations with different frequencies depending on the vehicle's speed. When passing through the same anomaly, not all vehicles produce the same longitudinal peak (depending on the vehicle's mechanical features). Driving speed also has significant effects on vehicle vibrations. Therefore, eliminating noise from the sensor data in order to smooth the signals and remove the speed dependency is critical [89, 93]. The main techniques of filtering and smoothing are outlined below:

Low-pass filtering: eliminate the high-frequency elements using a predefined frequency cut.

High-pass filtering: eliminate the low-frequency elements using a predefined frequency cut.

Band-pass filtering: A specific range of frequencies is used to pass some parts of the signals, while the other parts that are outside of that range are removed.

Moving average filtering: A low pass filter where information about the sensor data does not need to be known before.

Median filtering: comes in the forms of high pass and low pass filters where information about the sensor data does not need to be known before.

2.3.2.2 Resampling

Time series readings from motion sensors have to be obtained to produce road data. Unfortunately, IoT devices have a built-in problem with irregular and inconsistent cross timestamps, which prevents them from providing a regular sampling rate. The analysis of every detected event is significantly affected by data sampling rates. Therefore, we resampled the time series data at a regular rate, utilizing the resampling function to increase and decrease the sampling rate of the time series data [93].

2.3.2.3 Reorientation

The orientation of motion sensors affects the results of road anomaly detection. Consequently, reorientation is commonly implemented to transfer motion sensors data from the IoT device's coordinates to the coordinates of the vehicle in order to maintain the research's logic and compactness [94]. Figure 2.2 illustrates the orientation of motion sensors inside the vehicle.

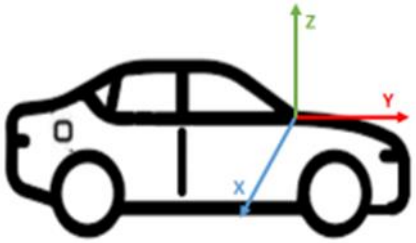


Figure 2. 2: Motion sensors orientation inside the vehicle.

To provide an approach that is not dependent on IoT device orientation, the n samples of motion sensors in which $\left(k_x(n);\ k_y(n);\ k_z(n)\right)$ are combined into a single magnitude:

$$m(n) = \sqrt{k_x(n)^2 + k_y(n)^2 + k_z(n)^2} \qquad (2.1)$$

In addition, Euler angles can be utilized to transform motion sensors' signals from IoT device coordinates to a different geometric coordinate.

2.3.2.4 Segmentation

The motion sensors vibration signals need to be split into segments by applying a sliding window to facilitate time series data analysis. Each segment should offer accurate data on the condition of the road surface. Hence, experiments are carried out in which the window size is varied in order to identify which size provides the best performance. Data segmentation step is essential since the convolutional layer is used as the input layer for classifier models [91].

2.3.2.5 Feature Extraction

Feature extraction techniques are thought to be advantageous for RSC monitoring since they allow for the extraction of more valuable data from each segment. The extracted features from road vibration signals are often generated using wavelet decomposition, frequency domain transformation, and time domain computation [81, 95] (see Table 2.3).

Table 2. 3: A presentation of the extracted features utilized in RSC monitoring.

Input features	Description	Feature types
Time-domain features	Extract statistical variables that automatically reflect the signals' vibrations.	-Standard deviation -Variance -Maximum value -Minimum value -Peak to peak -Root mean square -Mean of absolute value -Mean -Median
Frequency-domain features	Provides the frequency spectrum of the motion sensor signal by transforming it from the time domain to the frequency domain.	-Fast Fourier Transform (FFT) -power spectral density (PSD)
Wavelet decomposition	It is a frequency transform with the advantage of being defined in both spatial frequency and spatial location.	-Discrete Wavelet Transform (DWT) -Continuous Wavelet Transform (CWT)

2.3.2.6 Feature Selection

The features obtained from multiple domains are typically correlated, which may result in overfitting of the road surface anomalies detection classifier model. In many instances, feature selection comes after feature extraction in order to remove redundant features and enhance computing performance. To determine whether the features are correlated, we proposed three widely recognized correlation methods, namely Pearson, Spearman, and Kendall [96].

The Pearson correlation coefficient (r) is an index of the strength of the linear relationship between two different variables (a and b).

$$r = \frac{n(\sum ab) - (\sum a)(\sum b)}{\sqrt{[n\sum a^2 - (\sum a)^2][n\sum b^2 - (\sum b)^2]}} \qquad (2.2)$$

The Spearman's rank correlation (r_s) assesses both the strength and the direction of the relationship between two variables. Where D is the rank difference and m is a number of data pairs.

$$r_s = 1 - \frac{6\sum D^2}{m(m^2-1)} \qquad (2.3)$$

The equations above return values that range from -1 to 1, where:

- A value between 0 and ± 0.39 indicates a Low correlation
- A value between ± 0.4 and ± 0.59 indicates a Moderate correlation
- A value between ± 0.6 and ± 1 indicates a High correlation

Kendall's rank correlation (τ) offers an indicator of the strength of dependence based on the pattern of concordance and discordance among two variables.

$$\tau = \frac{2}{n(n-1)} \sum_{i<j} sgn(a_i - a_j)\, sgn(b_i - b_j) \qquad (2.4)$$

Kendall's rank correlation (τ) returns a value that ranges from 0 to 1, where:

- 0 indicates that there is no relationship.
- 1 is a perfect relationship.

2.3.2.7 Feature Scaling

The features extracted from road data have different distributions, resulting in a dominance of features with higher values and variance. Feature scaling addresses this issue by transforming the values of features to a similar scale, so that each feature is equally important. Normalization and Standardization are the two most commonly utilized methods for feature scaling [97, 98].

Normalization, often referred to as Min-Max scaling, is a method that shifts and rescales features with a distribution range between 0 and 1. Here's the general equation for Normalization:

$$f_{normalized} = \frac{f_- f_{min}}{f_{max} - f_{min}} \qquad (2.5)$$

The feature's maximum and minimum values are represented by f_{max} and f_{min} respectively.

Standardization, also known as Z-score normalization, is a method that rescales the features so that their standard deviation becomes 1 and their mean is 0. Here's the general equation for Standardization:

$$f_{standardized} = \frac{f_{raw} - \mu}{\sigma} \qquad (2.6)$$

Where σ and μ refer for the standard deviation and mean, respectively.

2.3.3. Classification

Monitoring the road condition is a classification problem in which segment features are the inputs and road surface types are the outputs. Deep learning (DL) is commonly utilized as a classification approach in RSC monitoring due to its ability to find patterns and relationships among inputs features and the labels of classes. However, recent approaches in human activity recognition (HAR) [99], medical image analysis [100], machine condition monitoring [96, 101], and electric energy forecasting domain [102] have combined different DL models to design hybrid deep learning models for enhancing the classification performance. The hybrid deep learning models that were utilized in this work are presented below.

2.3.3.1 Proposed CNN-LSTM and CNN-GRU Models

In this work, we proposed hybrid deep learning models that combine CNN with LSTM or GRU layers. The proposed architecture uses CNN to extract spatial features, whereas long-term temporal dependencies are extracted by LSTM/GRU layers [99, 101, 103]. In

this way, the CNN layers were used on the front end of the CNN-LSTM and CNN-GRU models for obtaining spatial features from the input data, while the CNN outputs were used by the LSTM/GRU layers for learning temporal features.

2.3.3.2 Proposed ConvLSTM Model

ConvLSTM is an LSTM variant that utilizes a convolution operation within the LSTM cell to detect spatial features in data. In comparison to CNN-LSTM, ConvLSTM is designed to utilize 3D data as input and reduce spatial data redundancy; it has achieved significant success in video frame classification. ConvLSTM captures fundamental spatial features from multi-dimensional data by using convolution operations at each gate in the LSTM cell rather than matrix multiplication [104, 105]. Figure 2.3 illustrates the structure of the ConvLSTM cell.

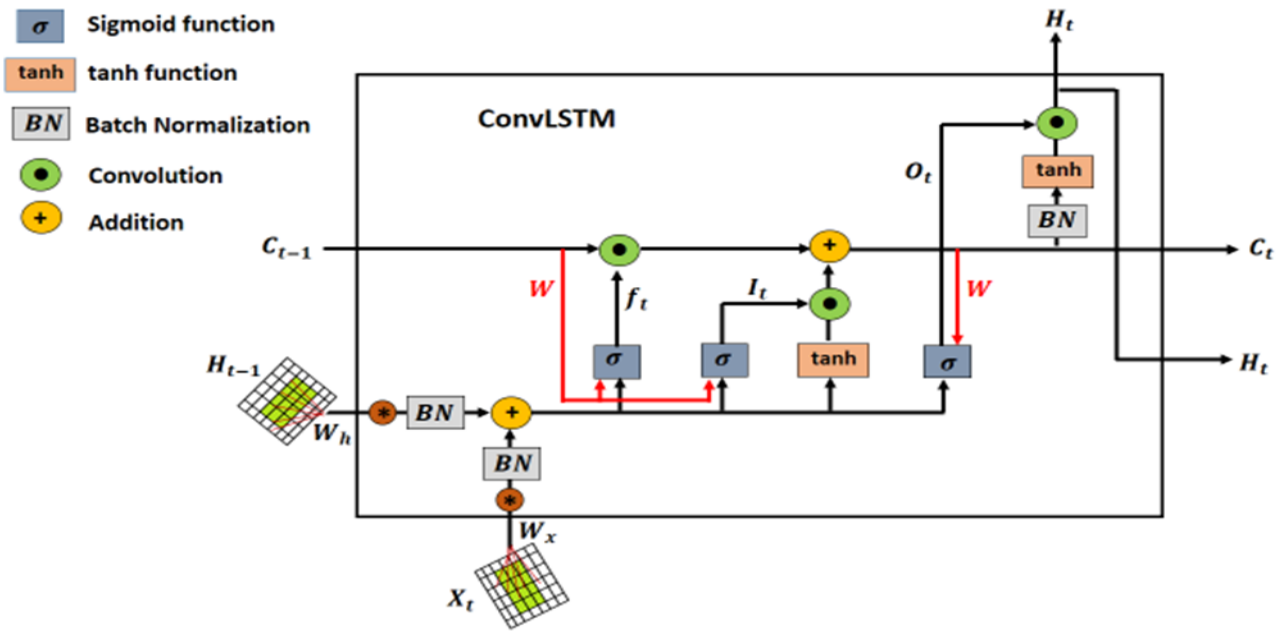


Figure 2. 3: ConvLSTM cell structure.

2.4 First Proposed Framework

The goal of this framework is to develop hybrid deep learning models for RSC monitoring, which employ a combination of FFT and DWT features as input. Indeed, sensor data from a smartphone's gyroscope, accelerometer, and orientation angles are used to make it easier to distinguish between road anomalies that cause two front wheels to hit

them at the same time and road anomalies that cause one front wheel to hit them at the first time. Also, we introduce a new technique for data labeling based on TCP/IP sockets that allow us to use a smartphone application to label sensor data in real time.

2.4.1. First Architecture

The first proposed framework's architecture is as follows: acquiring data related to road surface anomaly detection using motion sensors, preprocessing the collected data, dividing the dataset into train and test datasets, developing hybrid deep learning models for road surface anomaly classification, and evaluating the models through several evaluation metrics. Figure 2.4 depicts the architecture of the proposed framework.

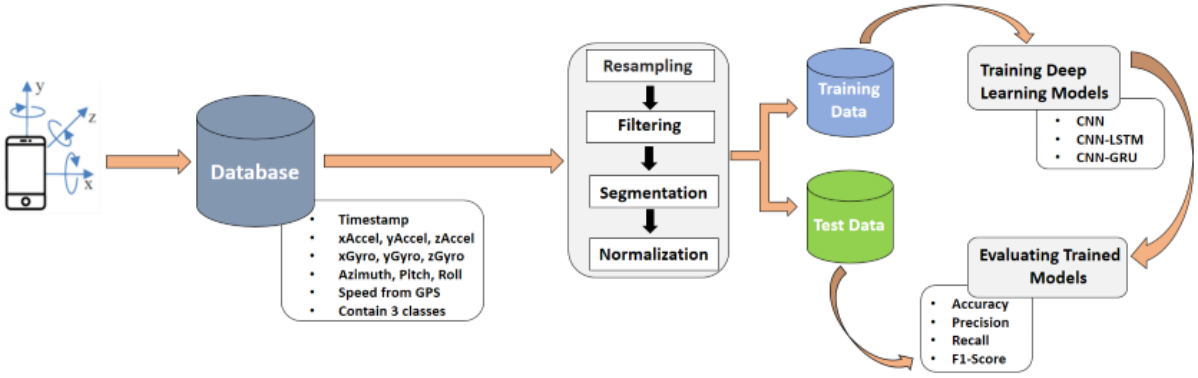


Figure 2. 4: The First proposed framework for road anomalies detection.

2.4.2. Data Acquisition

Our framework begins with the creation of the RSC dataset, which includes three different types of road surfaces: smooth road, man-made anomaly (speed humps and speed bumps), and real anomaly (potholes and cracks). A smartphone (Samsung, Galaxy J6+) was used to create the RSC dataset by employing accelerometer, gyroscope, and magnetometer sensors embedded into it. Further, we utilized the Android platform due to its status as the most common operating system for mobile devices [106]. After a thorough investigation, the GPS frequency sampling was fixed to 1Hz, while the sampling rate of the motion sensors was set to 50Hz. Inside the Renault Clio 4 vehicle, the smartphone was attached near the gear lever, where the y-axis was pointing forward and the z-axis was vertically orientated, while the x-axis was positive through the vehicle’s right side. A pair

of smartphones were utilized to collect data, as illustrated in Figure 2.5. The first smartphone was utilized to collect sensor data, while the other one was utilized for labeling the data in real time. When the first smartphone receives the road data label from the second smartphone (IRIS, Vox Fortis), it begins to send the device's current timestamp values, GPS data, and the three-axis motion sensors data to the database.

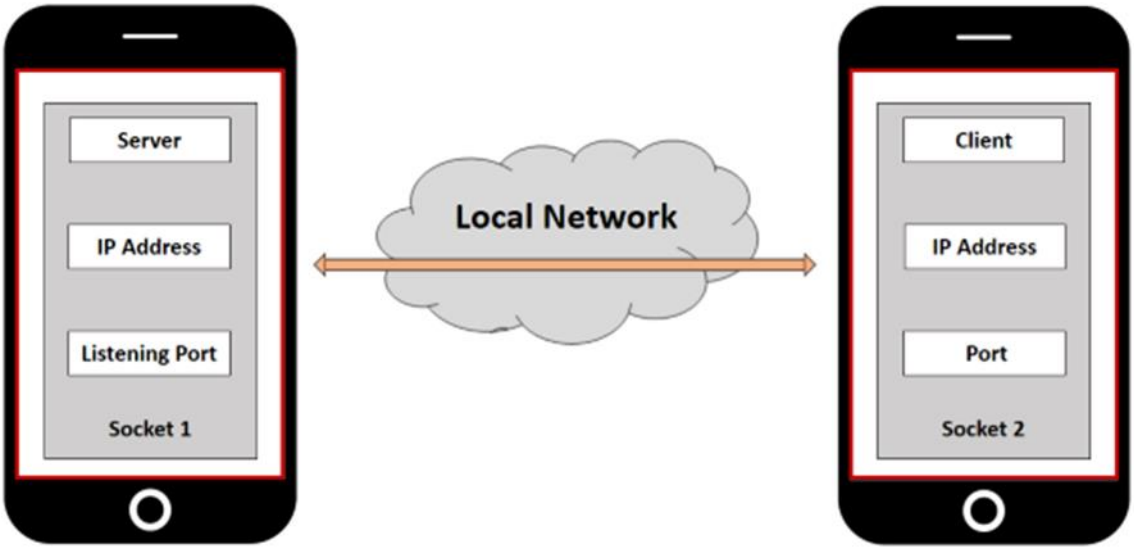


Figure 2. 5: Data labeling approach via TCP/IP sockets.

As shown in Figure 2.6, the two smartphones were communicating via a local network using a TCP/IP socket (client/server) [107]. Every smartphone has a socket, and each socket has been linked to a certain port number that it utilizes during the connection. In socket 1, a server runs on the first smartphone and just listens to socket 2 for a connection request from a client on the second smartphone. In socket 2, the client is aware of the first smartphone's IP address and port number, and it has to identify itself to the server. In the end, the road data label can be sent from socket 2 to socket 1 if the connection is established.

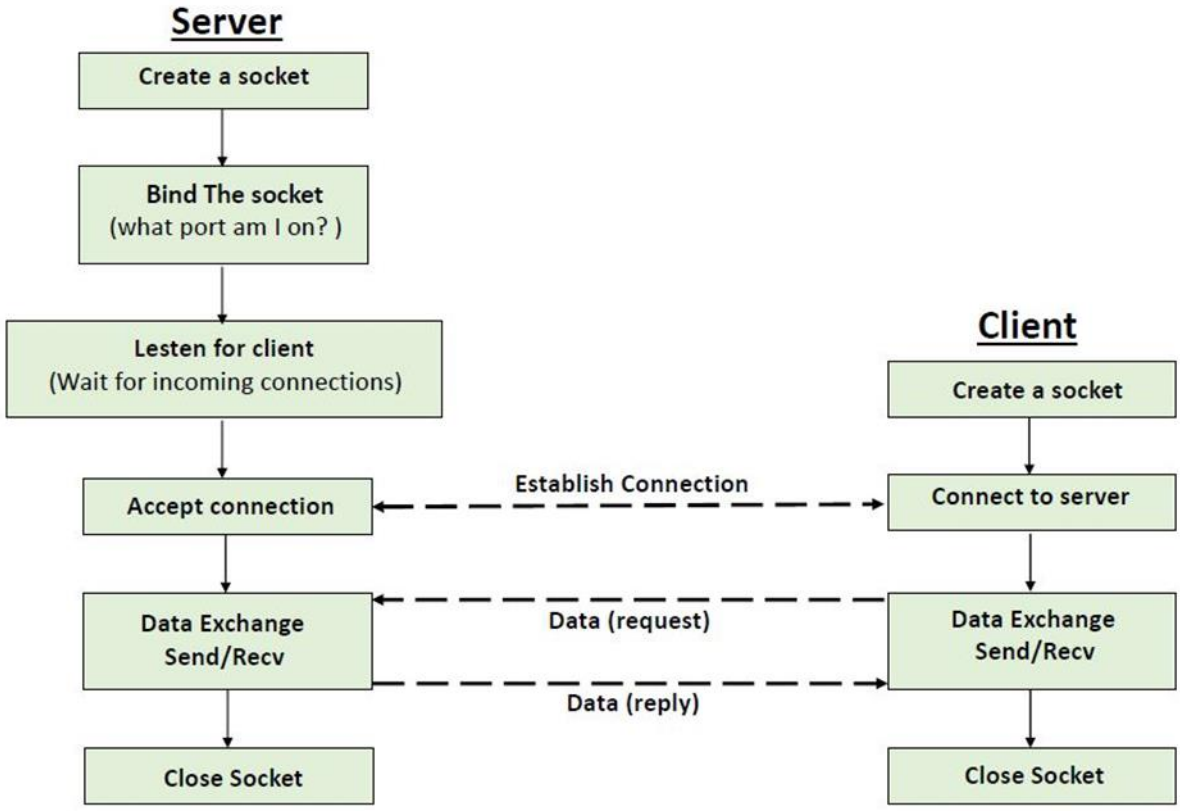


Figure 2. 6: TCP/IP Server–Client communication.

The RSC dataset was created when we were traveling from the University of Blida 1 to Boufarik city via four separate itineraries namely: "via Parallel N1", "via A1/N1", "via N29 and A1/N1", and "via Parallel N1". According to Google Maps, the distance is around 11-15 km. The first and fourth Google Maps itineraries have the same name but differing itineraries, reflecting the road trip from the University of Blida 1 to Boufarik city. During the ride, we collected data for classifying three types of roads: class 1 (pothole or crack), class 2 (speed hump or speed bump), and class 3 (smooth road). In total, we collected 1 114 062 data samples. All data were obtained by utilizing a Renault Clio 4 vehicle equipped with a MacPherson strut suspension, to ensure that the motion sensors measure in identical conditions on all itineraries. Table 2.4 shows the number of road anomalies, the distance, driving duration for each itinerary.

Table 2. 4: Distribution of road anomalies in the RSC dataset.

Google maps path name	speed bumps and speed humps	potholes and cracks	Distance	Driving Time
via Parallel N1	34	18	13.6 Km	40 min
via A1/N1	14	7	13.3 Km	18 min
via N29 and A1/N1	10	17	13.4 Km	33 min
via Parallel N1	35	16	11.4 Km	37 min

2.4.3. Preprocessing of Data

After data acquisition, time series measurements from motion sensors need to pass several steps of preprocessing in order to make the input data clear and simple. First, we resampled the time series data to 50 Hz because the motion sensors data sampling rate was irregular, as illustrated in Figure 2.7a. In the Android app, the sampling rate for the motion sensors was set to 50 Hz. However, Android is unable to preserve regular motion sensors sampling because the sensors sampling is prevented by the Android operating system and background apps, which take priority over system resources [108].

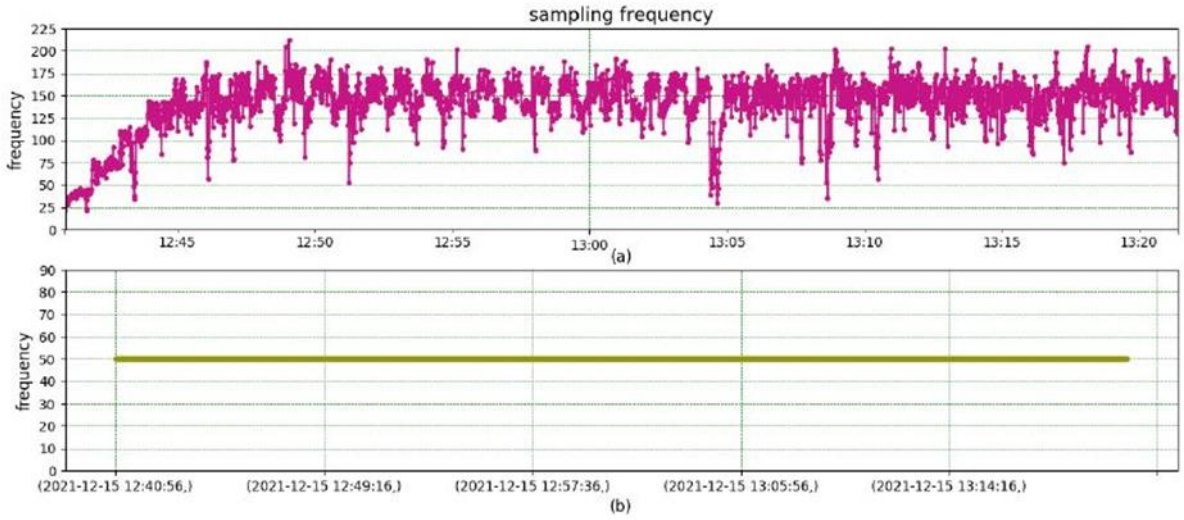


Figure 2. 7: Illustration of the resampling process showing (a) the original sensor sampling rate and (b) the new sampling rate.

As shown in Figure 2.7b, the irregular time series data of the original measurements was converted to a regular rate utilizing the resampling functionality. The missing values can

then be interpolated to this new rate via a polynomial interpolation technique that tries to fit a polynomial curve to these missing data values. By doing so, we got 388 250 data samples after resampling.

Second, data filtering is a prerequisite since motion sensors typically produce highly noisy signals, making the data unreliable. Therefore, we smoothed the sensor signals and eliminated noise using the moving average filter. A moving average filter is the simplest form of the low-pass filter which generates a new sample by averaging L samples from the input signal [109]. Here's the equation for the moving average filter:

$$s[i] = \frac{1}{L}\sum_{j=0}^{L-1} e[i-j] \qquad (2.7)$$

Where the average length is L, and the input and output signals are represented by e and s, respectively.

Here, we set the length to three, which means that the new sample is the average of the last two data points and the present data point. Figure 2.8 depicts z-axis acceleration data before and after the use of the moving average filter.

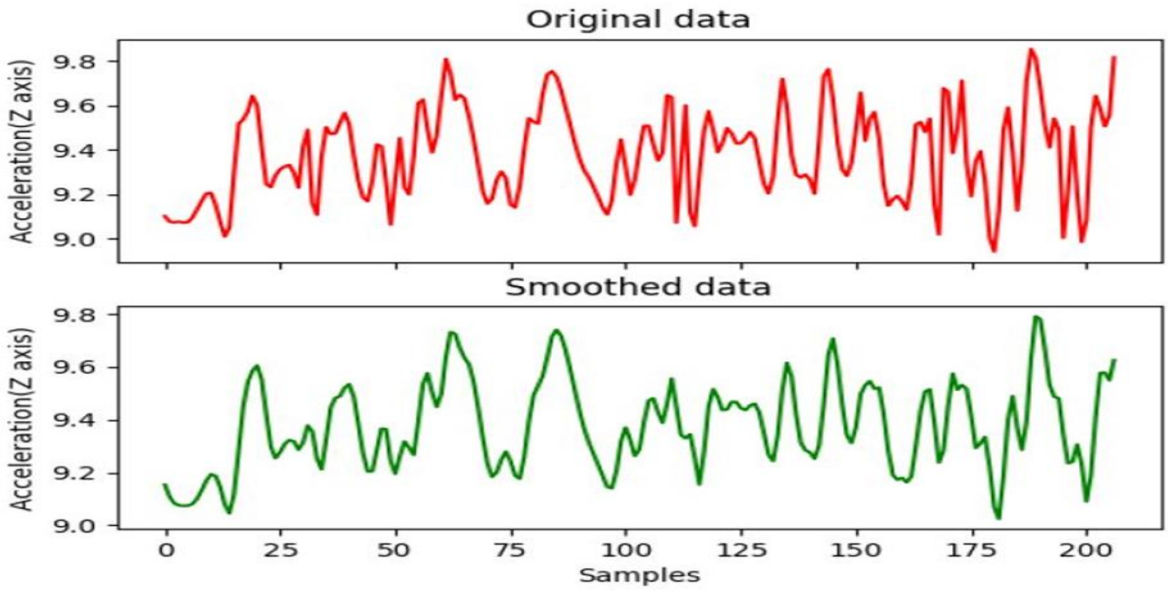


Figure 2. 8: Moving average filter of order L=3.

Third, the vibration data from the motion sensors were segmented using a sliding window of 1 second (50 samples), which is enough for representing any type of road anomaly when traveling at low speeds. In order to achieve this, we examined three overlapping factors: 66%, 50%, and 33%, which indicates that portions of the sensor's data will show in multiple windows, as can be seen in Figure 2.9.

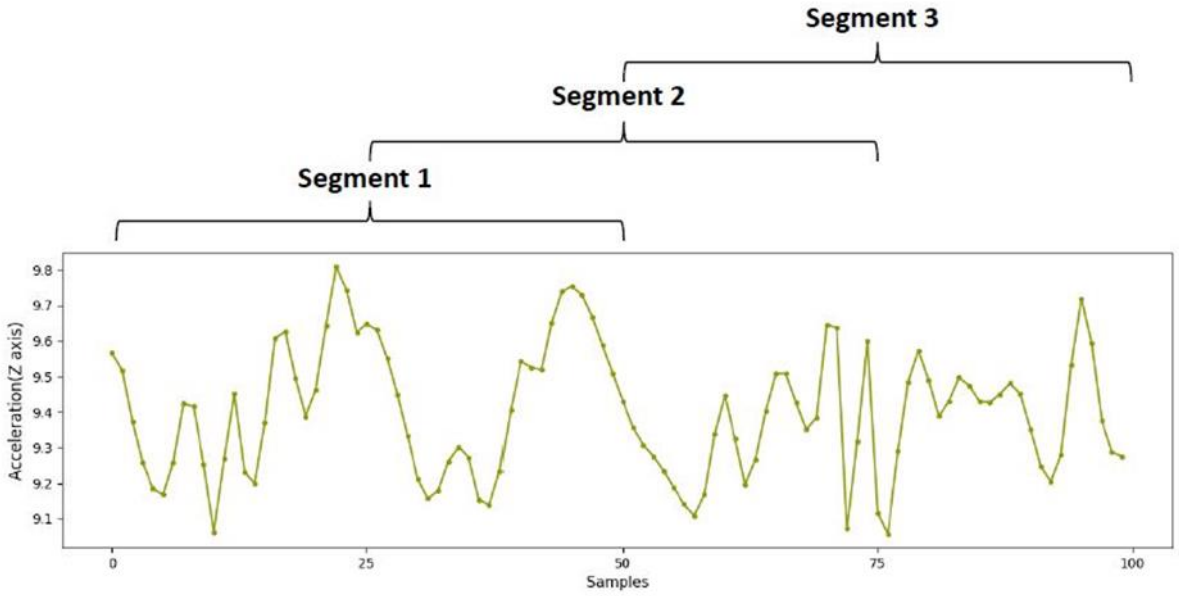


Figure 2. 9: Sliding window with 50% overlap.

Since some segments included samples with different road surface types (class labels), we deleted them all and labeled the remaining segments with the road data label that was present. When there was a traffic jam, we utilized the GPS data to delete smooth segments that had speeds below a certain threshold. Additionally, we used threshold-based filters to eliminate any segments with a road anomaly label and a speed above a certain threshold to deal with the effect of vehicle speed on motion sensors' signals. The distribution of the new segmented datasets is given in Table 2.5.

Table 2. 5: Distribution of the new segmented datasets.

classes	66% overlap	50% overlap	33% overlap
Smooth	6426	4105	3022
Man-mad anomaly	709	454	340
Real anomaly	675	434	321
All	7810	4993	3683

Fourth, the data were transformed using a multi-level 1D DWT combined with FFT to obtain information that is more useful from each segment, as illustrated in Figure 2.10. FFT provides spectral information, whereas the temporal location of the spectral components is given by DWT, that offers signal representations in the time-frequency plane. Specifically, we have employed the one-level coefficients of the Haar wavelet and the amplitude of FFT. The new segments have (50 × 9 wavelet coefficients + 50 × 9 FFT component amplitudes) = 900 samples. Compared to the original time-domain representation, these extracted features offer better classification performance.

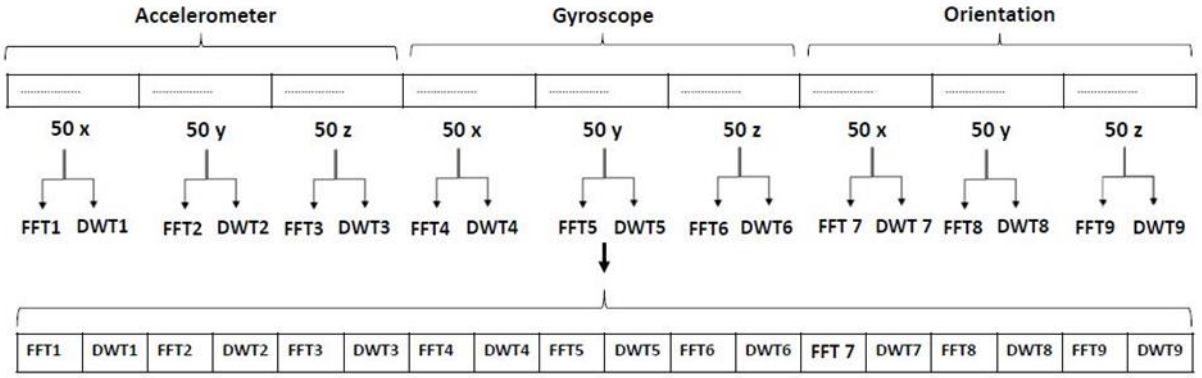


Figure 2. 10: Feature extraction using DWT and FFT.

Fifth, feature selection is carried out on segment data (DWT and FFT) to eliminate redundant features and keep only those that are important. From a random segment, we calculate the correlation coefficient between 50 wavelet coefficients and 50 FFT component amplitudes in the same sensor and axis (see Table 2.6). As can be seen, all correlation coefficients of Kendall, Pearson, and Spearman are lower than 38%, indicating that there is a low correlation between DWT and FFT features. So, we selected all features in the segment as input.

Finally, the features mentioned above have different value ranges. For instance, FFT provides a high amplitude at the first frequency and then drops to near zero. Every feature in the segment can be considered as noisy if its range of values is different. Consequently, the segment's features were standardized by subtracting the mean of every value and dividing by the standard deviation.

Table 2. 6: Correlation between DWT and FFT features.

Features	Kendall	Pearson	Spearman
DWT1+ FFT1	0.01731	-0.18856	0.02335
DWT2+ FFT2	-0.00577	0.10409	-0.01216
DWT3+ FFT3	0.09646	0.14134	0.13232
DWT4+ FFT4	0.18220	0.37227	0.23961
DWT5+ FFT5	-0.02226	-0.13672	-0.02450
DWT6+ FFT6	0.00412	0.02325	0.00879
DWT7+ FFT7	0.03050	-0.13746	0.05136
DWT8+ FFT8	0.01072	-0.10586	0.03186
DWT9+ FFT9	-0.00907	0.19077	-0.01936

2.4.4. Classifier Models

Here, the CNN-GRU, CNN-LSTM, and CNN classifier models were employed to evaluate the proposed framework. The CNN employed in the experiment was a multi-channel 1D CNN with convolution kernels that moved in one direction, as shown in Figure 2.11. The following were the 1D CNN parameter settings: the kernel size of the convolutional layers was (1 x 3), the pooling layer was set at (1 x 2), the activation function employed was ReLU, the optimization algorithm employed was Adam, and the dropout function was employed at a rate of 0.1.

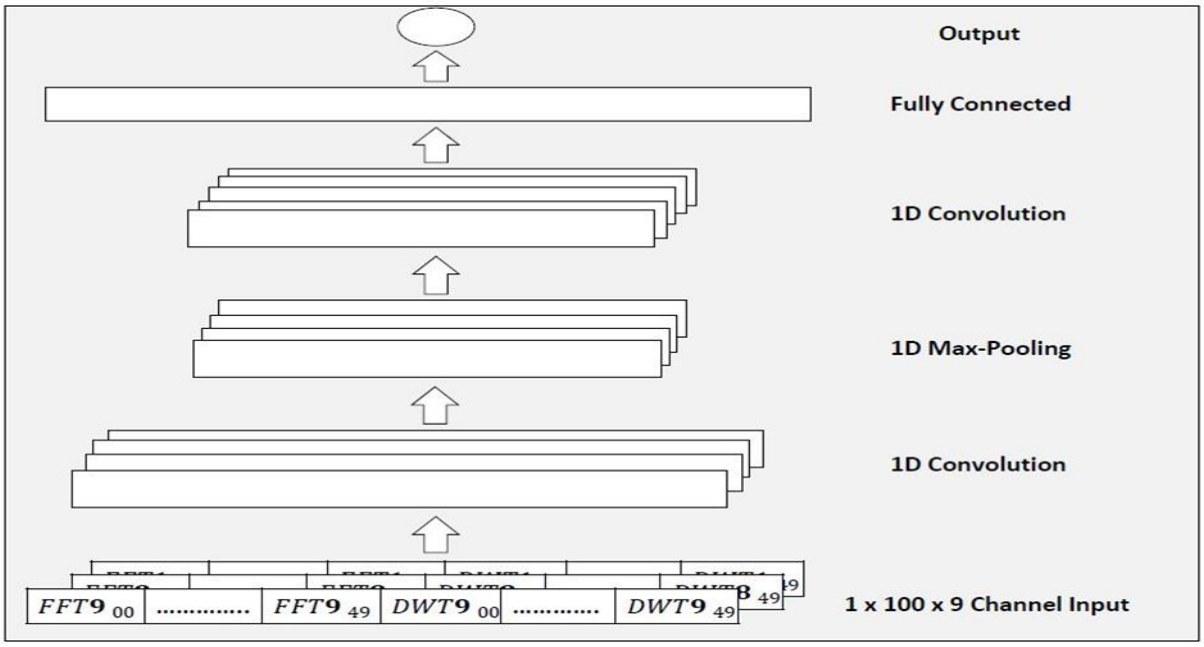


Figure 2. 11: Structure of a multi-channel 1D CNN model.

The hybrid deep learning models combined multi-channel 1D CNN with GRU or LSTM layers, in which the CNN features were inserted into the LSTM/GRU layers, as illustrated in Figure 2.12. This allows the combined model to swiftly determine spatial and temporal features, which can prevent the loss of some important data during CNN training [110]. Further, Table 2.7 provides the parameter settings for the CNN-GRU and CNN-LSTM.

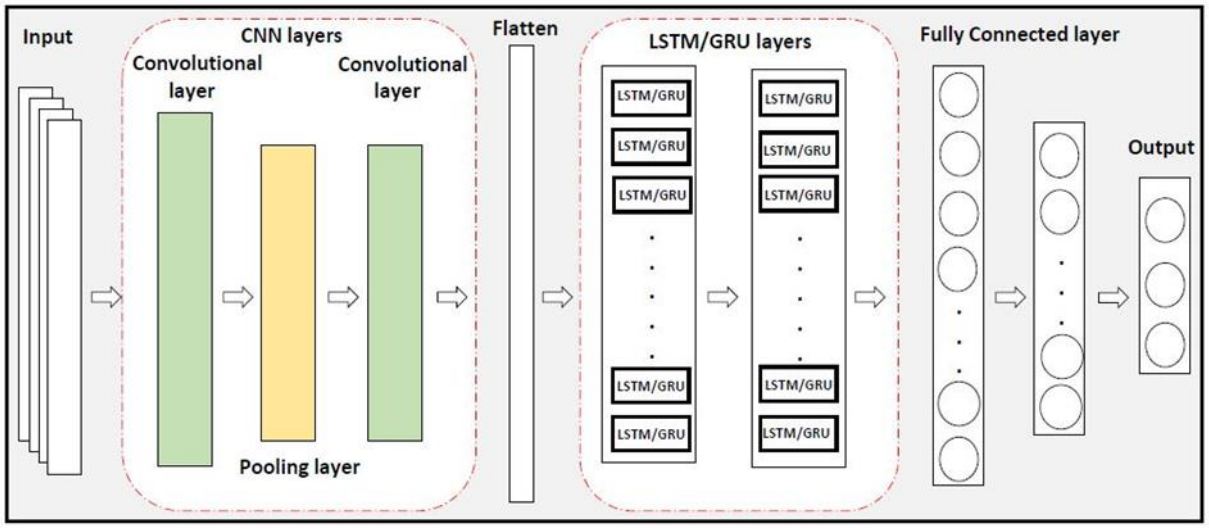


Figure 2. 12: The architecture of the proposed CNN-LSTM and CNN-GRU models.

Table 2. 7: The structure of the CNN- LSTM and CNN-GRU models.

Parameter	Parameter Settings
Input data size	1×100
Input channels	9
Convolutional layers	2 Conv2D
Filters	32-64
Filter size	[1 x 3] - [1 x 3]
Pooling size	[1 x 2]
LSTM/GRU	64-32
Dropout	0.1
Dense	128-64
Activation function	ReLU
Optimizer	adam
Dropout	0.1

2.5 Second Proposed Framework

The main idea of this framework is to develop 3D hybrid deep learning models to classify a vehicle's vibration data from an IoT device. This idea is inspired by video frame classification [111], which tests the effectiveness of spatiotemporal feature learning.

2.5.1. Second Architecture

The architecture of the second proposed framework is as follows: building a new RSC-IoT dataset and offering an explanation of the data labeling technique, performing a number of data preprocessing operations, and combining 3D hybrid deep learning models into two ensemble methods (see Figure 2.13).

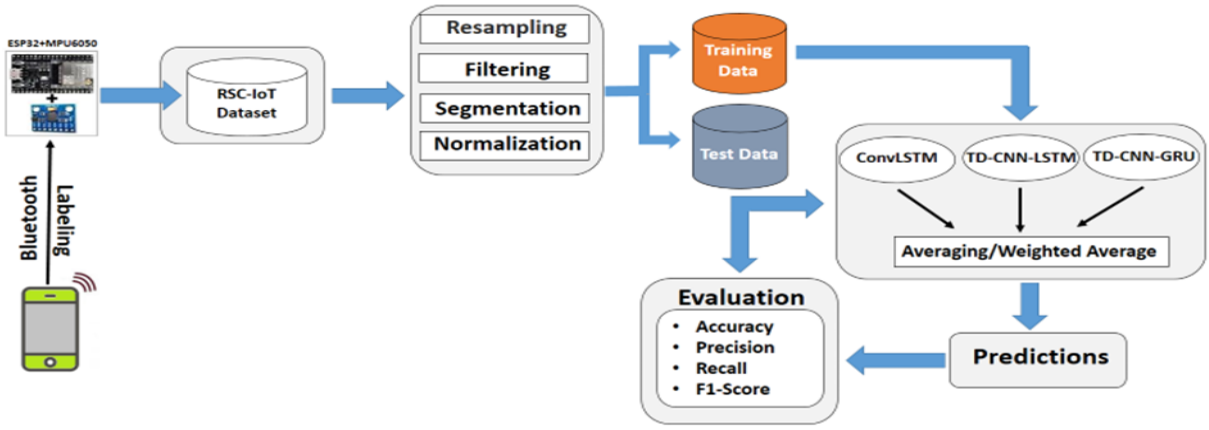


Figure 2. 13: The second proposed framework for road anomalies detection.

2.5.2. RSC-IoT Dataset

The RSC-IoT dataset was built for road surface condition monitoring, where a low-cost MPU-6050 with ESP32 microcontroller were used to record accelerometer and gyroscope data. Additionally, we used a smartphone that communicates with the ESP32 chip via Bluetooth to label raw sensor data in real time, as can be seen in Figure 2.14. The ESP32 chip, recognized for its integrated Wi-Fi and Bluetooth, is an excellent choice for IoT devices due to its low cost and power efficiency. During driving on three road surface types, namely "Man-made anomalies" (speed bumps), "Real anomalies" (potholes or cracks), and "Smooth" (flat road), the IoT device was attached to the top of the vehicle dashboard. Motion sensor data is transmitted to a local server using the HTTP protocol for IoT data storage.

Eventually, since every raw sensor data has a timestamp, 16669 data samples were acquired as time series.

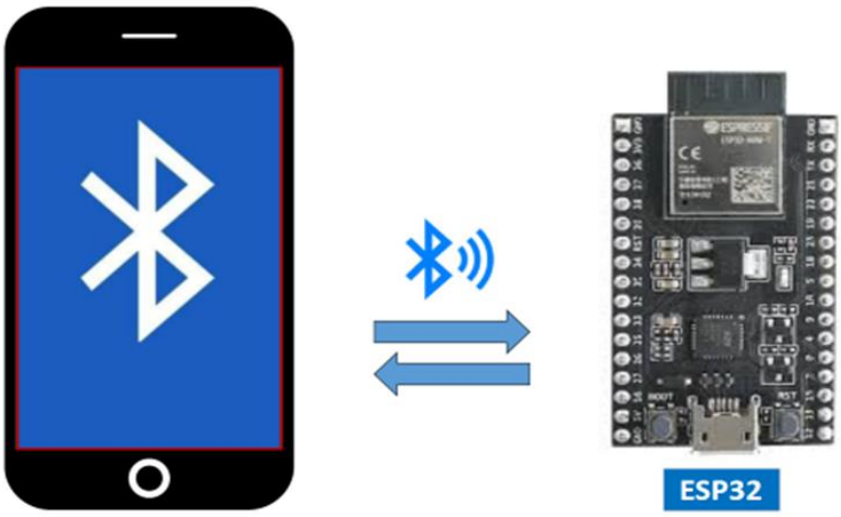


Figure 2. 14: Data labeling approach via Bluetooth.

2.5.3. Data Transformation

Four major steps were used in preprocessing to make the collected data clean and structured before the classification phase. At first, the time series data were resampled to 20 Hz because the motion sensors data sampling rate was below average (3-5 Hz). Secondly, in order to minimize noise in motion sensors data, the median filter [112] was used. Thirdly, a 2 second sliding window (40 samples) with a 66% overlap was used to segment the data, where every window has 240 components, as shown in Figure 2.15. Finally, in order to obtain a sufficient amount of samples from minority classes, the RSC-IoT dataset was manually split into train and test segments after the data segments had been standardized (see Table 2.8).

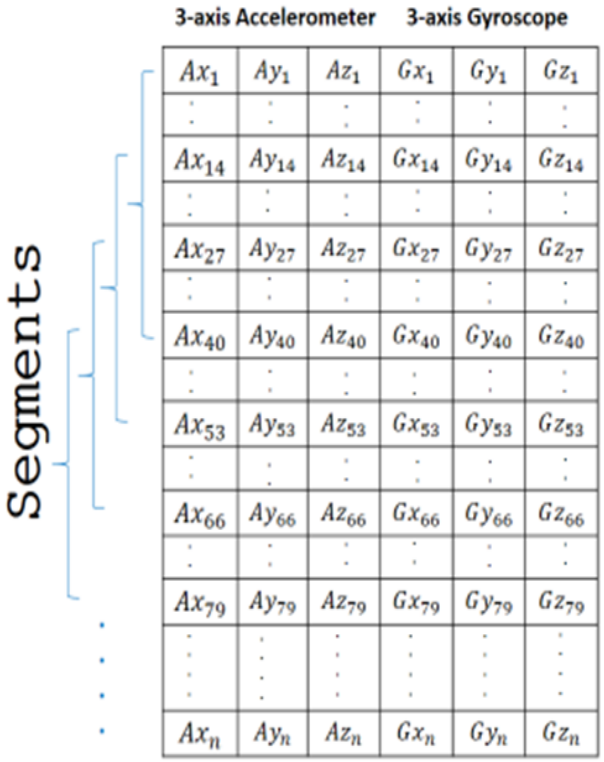


Figure 2. 15: Segmentation of sensors data with a 2 seconds and 66% overlap.

Table 2. 8: Distribution of data before and after preprocessing.

Step	Smooth		Man-made anomalies		Real anomalies	
Data collection	13139		783		2747	
Resampling	77473		6103		15607	
Segmentation	Train	Test	Train	Test	Train	Test
	4771	1006	313	107	918	176

2.5.4. The Hybrid 3D Models

Time series classification problems were well-suited for recurrent neural network (RNN) models such as LSTM and GRU because of their ability to identify long-term dependencies in the data. Since 3D models are typically used to learn spatiotemporal features from video frames, and as this study is inspired by video frame classification [113], both LSTM and GRU are not appropriate due to the one-dimensional format of their input data, which prevents the extraction of spatial features. To extract spatial features from multi-dimensional

input data, we built three hybrid deep learning models in which all CNN layers are wrapped by the TimeDistributed (TD) function [114]. The hybrid deep learning models are as follows: TD-CNN-GRU, TD-CNN-LSTM, and ConvLSTM. In order to provide a fair comparison between ConvLstm and standard CNN-LSTM results, the ConvLSTM model was not wrapped with a TD function. The structures of the three hybrid deep learning models are shown in Tables 2.9 and 2.10. The input was reshaped to the form (4, 10, 6, 1) before applying the 3D models, meaning that every frame of data represents 0.5 seconds of road data.

Table 2. 9: The structure of the ConvLSTM model.

Layer	Parameter Settings
ConvLSTM2D	filters: 32, filter size: [3 x 3], tanh activation
MaxPooling3D	Pooling size: [1 x 2 x 2], padding: same
Dropout	0.25
ConvLSTM2D	filters: 64, filter size: [2 x 2], tanh activation
Dropout	0.25
Flatten	-
Dense	n_neurons : 1024, relu activation
Dropout	0.25
Dense	n_neurons : 1024, relu activation
Dropout	0.25
Dense	n_neurons : 3, softmax activation, adam Optimizer

Table 2. 10: The structure of the TD-CNN-GRU and TD-CNN-LSTM models.

Layer	Parameter Settings
Conv2D	filters: 32, filter size: [3 x 3], relu activation
Conv2D	filters: 32, filter size: [3 x 3], relu activation
MaxPooling2D	Pooling size: [2 x 2]
Dropout	0.25
Conv2D	filters: 64, filter size: [3 x 3], relu activation
Conv2D	filters: 64, filter size: [3 x 3], relu activation
Dropout	0.25
Flatten	-
GRU/LSTM	units: 64, relu activation, return sequences: True
GRU/LSTM	units: 32, relu activation, return sequences: False
Dropout	0.1
Dense	n_neurons : 1024, relu activation

Table 2.10 (continued)

Dropout	0.25
Dense	n_neurons : 1024, relu activation
Dropout	0.25
Dense	n_neurons : 3, softmax activation, adam Optimizer

2.5.5. Ensemble Learning Methods

The 3D models were combined to perform the averaging and weighted average ensembles. Here, ensemble learning involves combining the outputs of the three 3D models to achieve more accurate predictions than the individual models.

2.6 Third Proposed Framework

The overall goal of this framework is to identify road surface anomalies using 3D hybrid deep learning models with synthetic time-series data to achieve peak performance. This framework is critical in demonstrating the benefit of data augmentation approaches in diversifying accessible information and keeping dataset balance.

2.6.1. Third Architecture

The third proposed framework shares the same architecture as the second, with the exception that it uses data augmentation to create synthetic time series training data for 3D hybrid deep learning models (see Figure 2.16), which enable the models to reliably and accurately identify anomalies in the road surface.

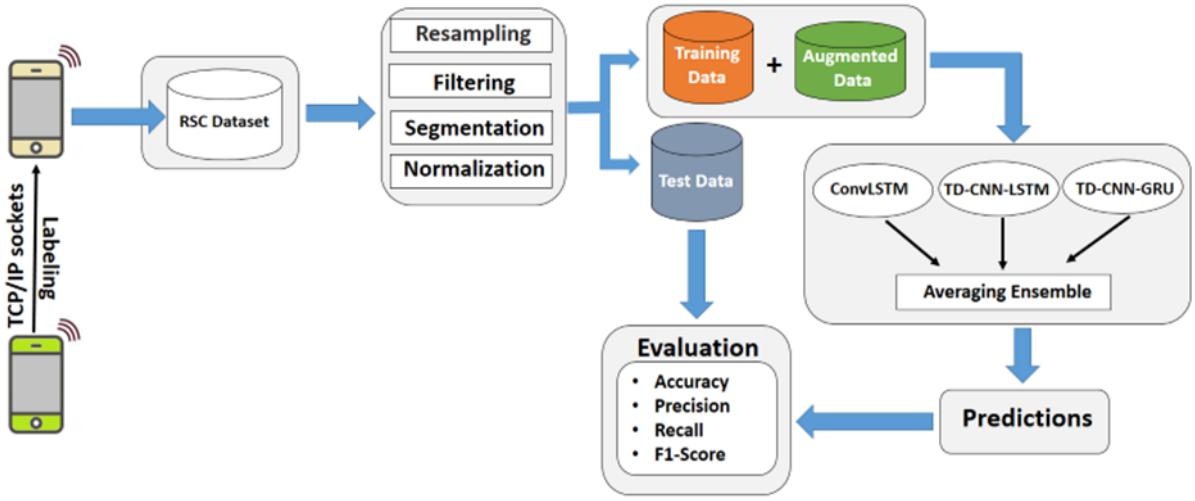


Figure 2. 16: The third proposed framework for road anomalies detection.

Based on the RSC dataset obtained from the first framework, we trained TD-CNN-GRU, TD-CNN-LSTM, and ConvLSTM models to classify road anomalies. The RSC dataset exhibits imbalance, as demonstrated in Table 2.11. Consequently, data augmentation approaches were employed to address the problem of imbalanced classification.

Table 2. 11: RSC dataset description.

classes	Road Anomalies	Number of Samples
Man-mad anomaly	Speed bump	50354
	Speed hump	
Real anomaly	Rough	41722
	Pothole	
Smooth	Smooth	1021986

In comparison to the second framework's data segmentation step, the data were segmented here using a sliding window of 3 seconds (150 samples) with 66% overlap. Each segment, as illustrated in Figure 2.17, consists of 3-axis accelerometer data, 3-axis gyroscope data, and orientation angles (1350 samples). After the segmentation, we obtained 2234 data segments, including 2028 segments of "Smooth" label, 133 segments of "Real anomalies" label, and 73 segments of “Man-made anomalies” label.

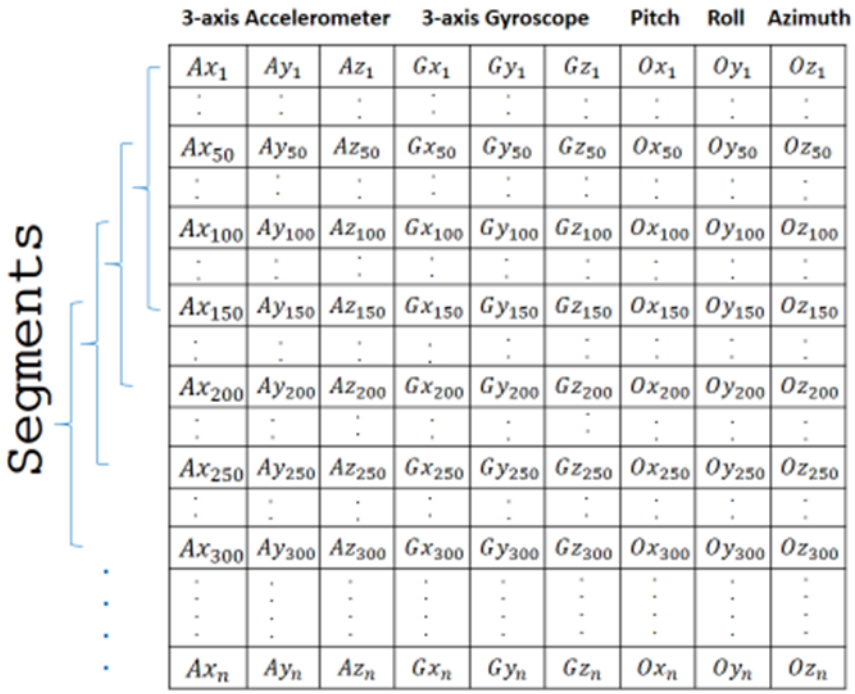


Figure 2. 17: Sensor data segmentation of 3 seconds and 66% overlap.

2.6.2. Data Augmentation

In RSC monitoring, it is well-known that driving over a road surface anomaly is an uncommon event, hence the quantity of speed bumps, speed humps, potholes, and cracks data is frequently considerably smaller than that of smooth roads. Data augmentation is an approach that handles the class imbalance problem by transforming the original data samples into synthetic data without affecting the data labels. Contrary to image data, it is challenging to confirm by human observation that the augmented data's label information has been preserved in the case of motion sensors data. Therefore, usually, the augmented data are only added to the training data to determine whether label information is preserved during data augmentation. To investigate the impact of data augmentation on road anomaly classification, classifier models were built in order to compare the results of models trained with the unbalanced and balanced training data using traditional time series data augmentation approaches[115-117], SMOTE algorithm [118, 119], and a GAN-based synthetic time series data generator called DoppelGANger [120].

Traditional data augmentation techniques have been successfully used to generate synthetic data in computer vision. However, changes caused by traditional time series data

augmentation techniques such as jittering, scaling, and rotation have no effect on the sensor data labels as shown in Figure 2.18. These techniques are as follows.

Jittering: Jittering is a technique that simulates additive sensors noise.

Scaling: Scaling is a technique that modifies the magnitude of data in a window through multiplying it with a random scalar.

Rotation: Rotation is a technique that simulates different sensor positions.

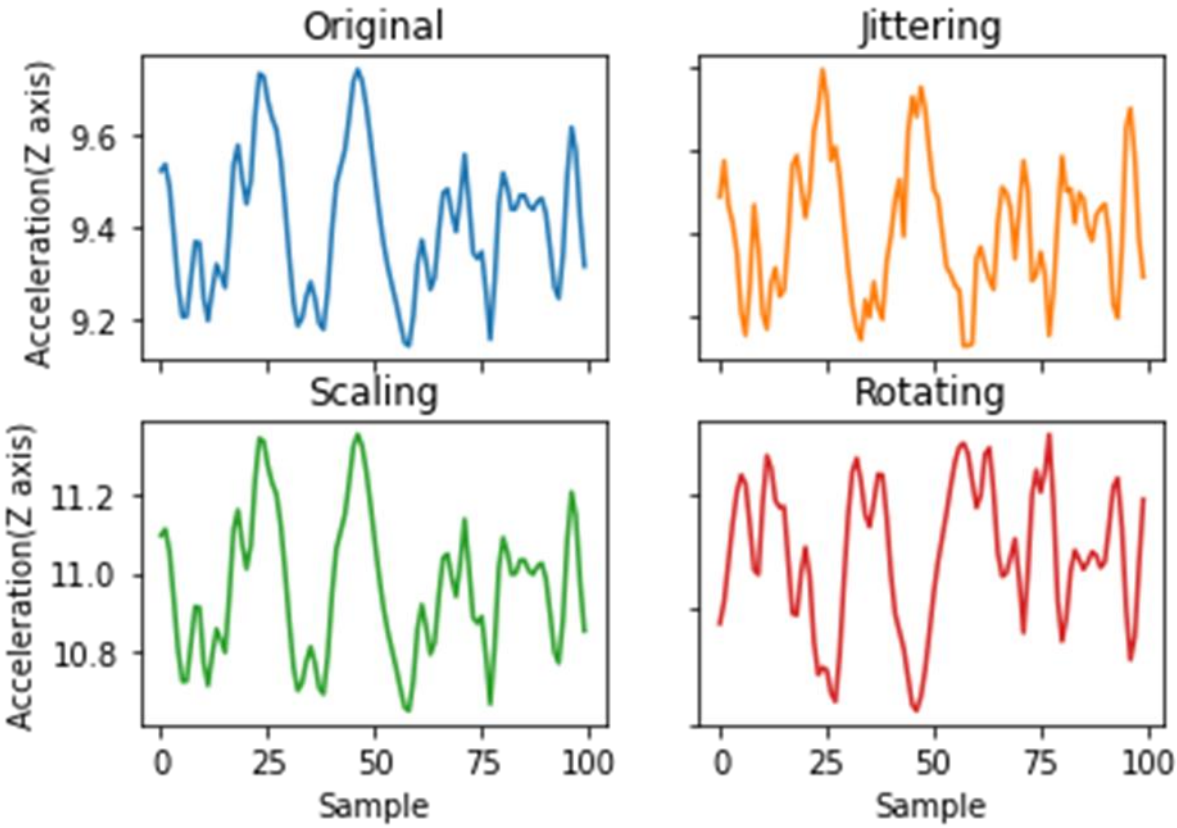


Figure 2. 18: Traditional time series data augmentation techniques.

SMOTE (Synthetic Minority Oversampling Technique) is a data augmentation technique that generates synthetic data for the minority class based on existing data points. The main idea behind SMOTE is to generate synthetic data between every minority class sample and its k nearest neighbors. The SMOTE algorithm works as follows:

- Selecting the data points from the minority class.

- Identifying the *k* nearest neighbors of the minority class data points.
- Generating new synthetic data points at random locations among all the neighbors until the problem of data imbalance is rectified.

Traditionally, generative adversarial networks (GANs) have had difficulty with the characteristics of time-series data. To overcome this problem, DoppelGANger was created by modifying a GAN to better suit the objective of time series data generation. The DoppelGANger's generator includes an LSTM to capture long-term temporal correlations and generate synthetic time series data, while the discriminator tries to differentiate between real and synthetic data [121].

For each of these data augmentation approaches, raw sensor data from the RSC dataset were selected randomly to generate synthetic data. Then, the augmented data were preprocessed, segmented, and combined with the training data for the classification phase. Table 2.12 shows the distribution of training data segments by road condition class labels.

Table 2. 12: Distribution of training data segments before and after data augmentation.

Data Aug. approaches	Real anomalies	Man-made anomalies	Smooth
No augmentation	89	51	1423
Traditional approaches	686	778	1423
SMOTE	836	798	1423
DoppelGANger	683	752	1423

2.7 Proposed RSC Monitoring Architecture using IoT Search Engine

As more smart cars are being used, more IoT data are being generated, making it difficult for RSC monitoring systems to find specific IoT data. Therefore, there is a need for an IoT search engine to provide query resolution services to aid RSC monitoring systems efficiently find relevant IoT data. The Internet of Things has had a significant impact on advanced manufacturing, healthcare, smart cities, and intelligent transportation. In intelligent transportation, smart cars are equipped with IoT devices. These IoT devices contain sensors such as cameras, GPS, proximity sensors, vibration sensors, etc, which detect changes in the external physical world and record them. Thanks to GPS sensor,

smart cars are capable of offering the shortest route to the destination, as well as anticipated traffic and travel time. However, since it depends on the preferences of the drivers, such as comfort and ease of driving, the best path may not always mean the shortest trip time or distance. The optimal path is defined as the one with the shortest trip time, the shortest travel distance, and the fewest road surface anomalies. This means the road condition should be considered as an additional factor in determining the best path. In this work, we focused on the design of an IoT-based road surface condition monitoring system that continuously provides road anomalies locations for drivers that prefer the path with the fewest road surface anomalies. Figure 2.19 depicts the overall workflow of the proposed IoT-based RSC monitoring system using the IoT search engine (IoTSE).

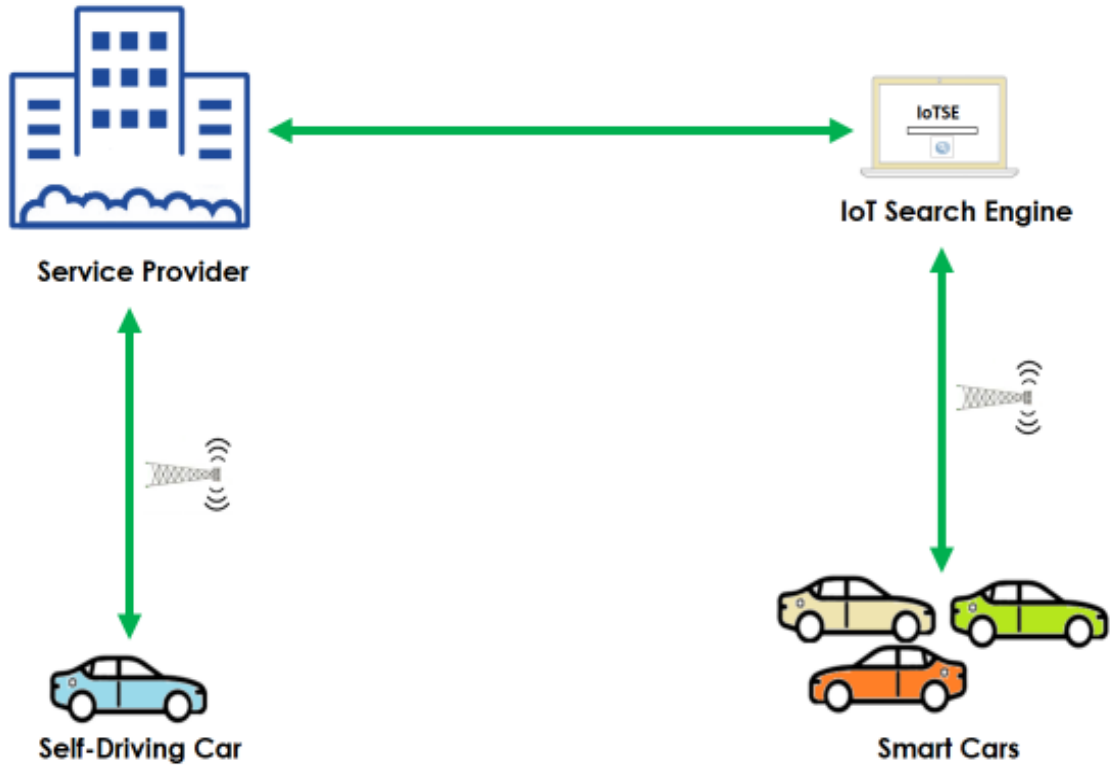


Figure 2. 19: The architecture of the proposed RSC monitoring using IoT search engine (IoTSE).

The proposed system recommends self-driving cars to the destination using the path with the fewest road surface anomalies. Our framework of the IoT search engine (IoTSE) is inspired by the studies [74, 75] and is as follows:

Step 1: A self-driving car sends queries to a third-party service provider to find the path with the fewest road surface anomalies.

Step 2: The third-party service sends again the queries to the IoT search engine (IoTSE).

Step 3: Based on the user's location, the IoTSE utilizes an IoT crawling algorithm to collect GPS and vibration sensors data from IoT devices embedded in smart cars on the road.

Step 4: The sensor's data are again transmitted to the third party service provider for collecting and preprocessing.

Step 5: The third-party service provider detects road surface anomalies using pre-trained hybrid deep learning models.

Step 6: Finally, the self-driving car can find the most convenient path to the destination with the fewest road surface anomalies.

2.8 Proposed RSC Monitoring Architecture using Cloud Computing

The trend of combining cloud computing and IoT technology has gained popularity in the past decade. Cloud computing is a component that helps the Internet of Things succeed by addressing the problem of collecting and analyzing sensor data that can be utilized later to monitor any system. The overall workflow of the proposed IoT-based RSC monitoring system using cloud computing is detailed in Figure 2.20.

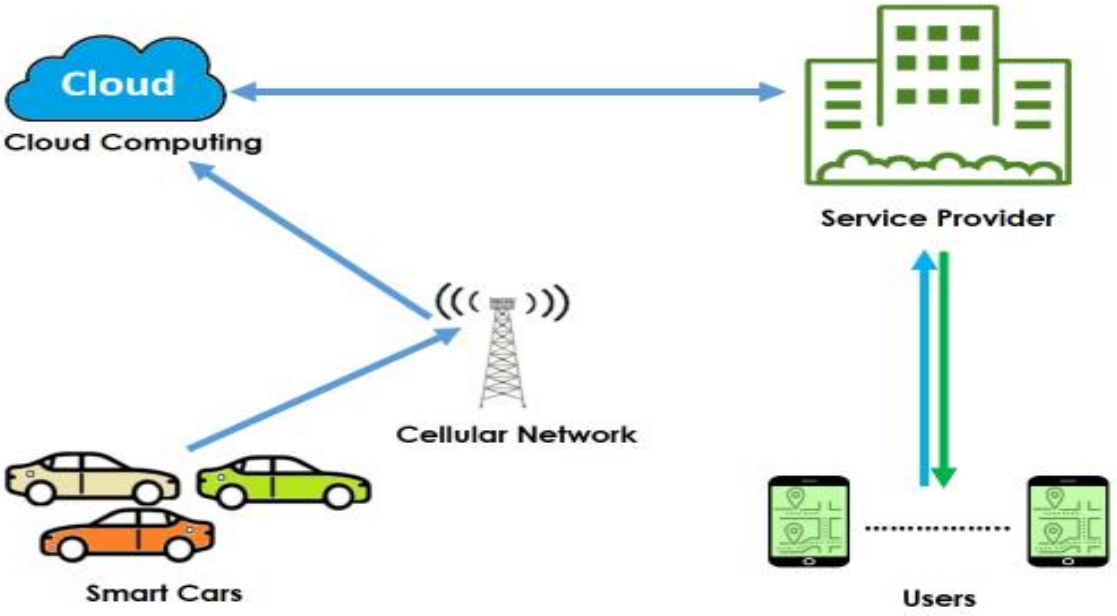


Figure 2. 20: The architecture of the proposed RSC monitoring using cloud computing.

The proposed system provides drivers with the geographical location of detected road anomalies. This work has several steps that can be summarized as follows:

Step 1: GPS and vibration sensors data are continuously captured by IoT devices embedded in smart cars on the road.

Step 2: The data are transmitted to the cloud computing platform over the internet via cellular networks.

Step 3: The sensors data are continuously received in the cloud for collecting and preprocessing.

Step 4: The preprocessed data are again transmitted to a third party service provider.

Step 5: The third-party service provider detects road surface anomalies using pre-trained hybrid deep learning models.

Step 6: Finally, using smartphones or tablets, drivers are able to show the geographic location of the detected road surface anomalies on a street map offered by the third-party service provider.

2.9 Conclusion

In this chapter, we presented road surface condition monitoring systems that detect road anomalies using hybrid deep learning models. The systems are based on motion sensors from smartphones and IoT devices. These systems include data collecting, preprocessing, and classification, where the vibration-based method is used to collect time series data.

Further, two datasets, RSC-IoT and RSC dataset, were created and labeled in real time utilizing a smartphone via Bluetooth and a TCP/IP socket, respectively. We also proposed RSC monitoring architectures based on cloud computing and the IoT Search Engine (IoTSE).

CHAPTER 3: BIG DATA AND MACHINE LEARNING FOR IOT INTRUSION DETECTION SYSTEM

3.1 Introduction

The Internet of Things (IoT) is an innovative framework for the future internet in which interconnected structures of devices connect to the internet in order to enable smooth data exchange. IoT devices can be controlled and monitored remotely. Some of these devices include smart washing machines, self-driving cars, smart toothbrushes, industrial robots, smart lights, medical and healthcare equipment [122]. Due to the extensive growth of IoT networks and the constantly expanding applications they support, hackers are now able to target these networks with a variety of security threats [123].

Intrusion detection systems (IDS) served as an essential means for protecting information systems and IoT networks for many years. IDS includes software or hardware that detects unlicensed usage on IoT systems. Through monitoring all traffic that enters and exits the network, the IDS is a highly effective tool for network security. IDS technology has developed for traditional networks, however existing solutions are insufficient for IoT since they are not sufficiently adaptable to handle the diverse and complicated IoT environment [124].

In fact, artificial intelligence (AI) techniques could be employed to solve the need for the creation of advanced security technologies adapted to the IoT. These days, intrusion detection systems (IDS), have been using ML and DL algorithms as novel tools to improve their effectiveness in securing IoT networks. In addition, efforts have been made to enhance threat detection and avoidance with ML and DL based solutions, with a focus on identifying dangerous IoT traffic flow through pattern recognition of ordinary and irregular traffic during the training stage [125, 126].

Big amounts of data are generally generated by IoT devices and are transmitted over networks. Network attacks might modify data that flows via a network. Combining big data and artificial intelligence technology can help secure the IoT network by identifying and classifying attacks [70].

A variety of datasets are employed in the fields of IoT security, big data, and the experimental research of ML and DL. IoT security can be described as protecting the IoT network infrastructure and its components from attacks. The tools required for efficiently analyzing data that have been defined as big data are known as big data technologies. Further, to process IoT network data, algorithms based on ML and DL are useful in order to identify zero-day attacks and anomalies in network traffic [125].

IDS experiments have been lately conducted using numerous publicly available IoT datasets. Nevertheless, there is an underlying class imbalance problem with these datasets that can lead to low performance in minority classes. Various data sampling approaches, such as oversampling, undersampling, and ensemble resampling, are typically employed since achieving high accuracy in detecting intrusions requires balancing the dataset [127, 128].

In this chapter, we introduced a big data architecture that applies ML to perform multi-class classification on an imbalanced dataset for monitoring IoT network traffic. The primary objective of this study is to address the issue of class imbalance and enhance classification performance through the use of an ensemble resampling technique in conjunction with oversampling. Furthermore, the final predictions are computed utilizing the ensemble learning methodology.

3.2 Literature Review

In the field of computer science, there has been a lot of interest in the research and development of intrusion detection systems (IDS) due to the growing number of networks and security threats. The IDS is a tool that analyzes network traffic to identify any suspicious patterns or known attacks. This section covers the main approaches and selected datasets utilized in IDS solutions. Also, some recently published papers on IDS publications are presented.

3.2.1. IDS Approaches

In the context of IoT security, various attacks against IoT networks have been discovered in the literature. In order to identify these attacks, several features can be

extracted from the network traffic flow and stored in a dataset [129]. In general, three main types exist for intrusion detection approaches [130]:

- The signature-based detection: is based on databases that comprise patterns or attack signatures to detect well-known attacks.
- The anomaly-based detection: analyzes the behavior of users, networks, and system hosts which then alerts the administrator whenever the behavior deviates from the expected behavior.
- The hybrid-based detection: combines the signature-based detection approach with the anomaly-based detection approach.

However, each intrusion detection approach has strengths and limitations [122, 131]. A summary of intrusion detection approaches' pros and cons is provided in Table 3.1.

Table 3. 1: The advantages and disadvantages of intrusion detection approaches.

Detection methods	Pros	Cons
Signature-based detection	-Easiest and more successful in identifying known attacks. -Design is simple. -Identifies the intrusions quickly. -Highly effective in detecting intrusions with minimal false alarms.	-Incapable of identifying novel attacks and variations on known attacks. -New signatures should be added on regularly. -Inappropriate for identifying multi-step attacks.
Anomaly-based detection	-Efficient at finding unexpected and novel vulnerabilities. -Finding new attacks is the main goal of it. -Make it easier to identify incidents of privilege abuse. -Possibility of generating an intrusion signature.	-Initial training is required. -Setting off alarms at the proper time is challenging. -There are a lot of false positive alarms. -Not classified alerts.
Hybrid-based detection	-Recognize both known and unknown attacks. -Reliability and confirmation of alerts.	-Increase the complexity and resource requirements of the system. - Detecting attacks takes quite a while.

Security technology has recently seen a trend toward the employment of popular ML and DL classifiers for building IDS to monitor IoT network traffic. The goal is to use

binary and multiclass classification to detect cyberattacks. These classifier algorithms analyze and detect patterns in IoT network traffic data. In fact, a huge quantity of data is continuously produced by IoT devices which are used more and more in our daily lives. Thus, big data processing is required in order to extract useful information from such data. Because some data are imperfect, data cleaning, transformation, normalization, and dimensionality reduction are frequently performed to reduce the complexity of data and minimize processing time [132]. In order to improve final predictions, ML or DL algorithms are often combined using ensemble learning methods [133, 134].

3.2.2. IDS Datasets

The hackers aimed to utilize malicious activities to weaken the resources of the targeted IoT network, necessitating the usage of intrusion detection systems (IDSs). In order to train and evaluate IDS in IoT networks, anomaly detection datasets are required, which could also be used for the evaluation of ML and DL algorithms' performance. However, because there are insufficient organized IoT datasets for intrusion detection, IoT networks are challenging to analyze and evaluate. The following is a description of some popular datasets that contain IoT traffic:

MQTTset: The dataset MQTTset was created by Vaccari et al. [135]. This dataset contains 541,071 instances for training and testing. In order to simulate a smart IoT environment, the MQTT protocol was used to communicate between multiple IoT devices. MQTTset dataset class categories consist of SlowITe, Malformed Data, MQTT Publish Flood, Flooding DoS, Bruteforce Authentication, and Normal.

MQTT-IoT-IDS2020: The dataset MQTT-IoT-IDS2020 was created by Hindy et al. [136]. This dataset has 3,654,006 data instances. The most typical MQTT attacks are included, along with scenarios for real-life tests equipment. The class categories of the MQTT-IoT-IDS2020 dataset include Sparta, Scan-U, Scan-A, MQTT-Bruteforce, and Normal.

IoT Network Intrusion: The IoT Network Intrusion dataset was generated by Kang et al. [137]. This dataset was processed using a typical smart home design that included an EZVIZ Wi-Fi camera and a smart home SKT NGU. There are 625783 data instances in the

IoT Network Intrusion dataset. It comprises the following class categories: Normal, Scanning, Mirai Botnet, MITM, and DoS.

IoT-23: This dataset is based on network traffic that was collected by IoT devices using three benign and twenty malicious captures. The Stratosphere Laboratory of Czech Technical University (CTU) created the IoT-23 dataset. The IoT-23 dataset includes 10 class categories and 106,542,182 data instances [138].

TON_IoT: The TON_IoT Telemetry dataset is the next generation of IoT and Industrial IoT datasets designed to evaluate the effectiveness and accuracy of various AI-based cybersecurity systems. This dataset was provided by Moustafa et al, IoT Laboratory at UNSW Canberra, Australian Defence Force Academy (ADFA) [139].

UNSW-NB15: The UNSW-NB 15 dataset was produced using the UNSW Canberra Cyber Range Lab's IXIA PerfectStorm tool. The aim is creating a combination of fake contemporary attack behaviors and realistic modern daily activities. This dataset contains 175,341 instances for training and 82,332 instances for testing [140].

BoT-IoT: The BoT-IoT dataset will be described in detail in the next section.

3.2.3. A Review of IDS Studies

A substantial amount of research has been done in recent years on the Internet of Things security. The security of the IoT is more susceptible to vulnerabilities and attacks. IoT security is entirely dependent on the functionality of the Intrusion Detection System (IDS) [141]. Table 3.2 provides an overview of some recent IDS studies that made use of IoT traffic datasets.

Table 3. 2: Summary of studies that used IoT traffic datasets.

Reference	Datasets	Detection Approach	Contributions
Saheed et al. [142]	UNSW-NB15	Signature-based	Applying an intelligent combination of feature dimensionality reduction and ML techniques, which created an intelligent IDS that can identify suspicious behavior on insecure IoT networks.
Saba et al. [143]	BoT-IoT	Anomaly-based	Presenting an anomaly-based IDS based on CNN that exploits the potential of the IoT by offering capabilities to effectively analyze all traffic passing through it.
Albulayhi et al. [144]	IoTID20	Anomaly-based	-Developing a hybrid feature selection method by utilizing the theory of mathematical sets. -Combining a majority voting system and a variety of ML methods to build an intelligent intrusion detection system that achieves the highest rate of detection

Table 3.2 (continued)

Simon et al. [145]	NSL-KDD	Signature-based	Employing a classifier that combines CNN and DT algorithms to identify attacks in the Internet of Things network.
Sahu et al. [146]	IoT-23	Signature-based	-Employing a hybrid deep learning model called CNN-LSTM to identify attacks and monitor IoT networks. -A comparison of the CNN-LSTM model with other recent, related research studies.
Zhang et al. [147]	CSE-CIC-IDS2018 CIC-IDS2017 NSL-KDD	Signature-based	Applying the ICVAE-BSM method to efficiently identify minority intrusions from unbalanced IoT samples.
Ferrag et al. [148]	TON_IoT MQTTset Bot-IoT	Signature-based	Developing federated learning models using three DL algorithms, which are RNN, CNN, and DNN to provide more accurate results and assure the privacy of IoT device data.
Demirpolat et al. [149]	UNSW-NB15 Bot-IoT	Signature-based	Software defined networking (SDN), which used a few-shot learning classifier to address the problem of building ML models with few training samples.

Table 3.2 (continued)

Ramadan et al. [150]	NSL-KDD	Anomaly-based	- Presenting a hybrid IDS for detecting IoT network attacks. - Using the ESFL algorithm for feature extraction. -Applying a hybrid classification algorithm called the LCNN-GRNN.
Abushwereb et al. [151]	Bot-IoT	Signature-based	-Using Apache Spark to create a big data framework for classifying IoT network intrusions. - Utilizing the entire dataset as well as the shorter version of the BoT-IoT dataset.
Manzano S et al. [152]	Bot-IoT	Signature-based	-Developing a Hadoop-Spark cluster-based platform for big data analysis and processing. -Using One-Class SVM to assess if the flow of traffic is malicious or benign, and an RF Multi-class model to define the attack type.

Table 3.2 (continued)

Manzano et al. [153]	Bot-IoT	Signature-based	-Introducing a big data architecture that employs Hadoop-Spark for classification of multiple classes using a one-vs-rest technique. -Evaluating the effectiveness of three oversampling techniques: CTGAN, SMOTE, and ADASYN in producing additional instances from minority classes.

This study aims to employ a signature-based detection approach for IoT intrusion detection using the full BoT-IoT dataset in the Apache Spark environment. In fact, a number of solutions have been created to deal with the issue of imbalanced classification in the BoT-IoT dataset. As far as we are aware, there are just three researches [151–153] that employed multi-class classification using the whole BoT-IoT dataset in a big data environment. In this case, we perform more accurate classification studies in a big data environment using the BoT-IoT dataset. Our methodology is original in comparison to previous studies because it employs a novel technique that combines ensemble resampling and oversampling to address the issue of class imbalance. Additionally, in order to achieve superior prediction results, we combine n separate ML models built on Apache Spark using a weighted average ensemble.

3.3 Proposed Methodology for IoT Network Intrusion Detection

The proposed intrusion detection methodology is described in extensive detail here. The first step in this process is to load the BoT-IoT dataset, following which data preprocessing is necessary to put the data in a format that is easier for analysis. The next phase involves using an ensemble resampling method in combination with oversampling to

address the problem of the imbalanced dataset. The Decision Tree ML algorithm is then employed in order to identify patterns in network traffic that could lead to the detection of attacks on the data features. It is essential to implement forensic analytics that employ big data and ML. After that, the ensemble learning method is used to incorporate the predictions for obtaining the final result. This approach has similarities to the bagging method, with the exception that we manually pick the data points and resample the data via oversampling, and the weighted average ensemble is used to obtain the final predictions instead of selecting random data with replacement then incorporating the results using the averaging ensemble. However, data analysis is done using a Big Data environment called Apache Spark because of the size of the BoT-IoT dataset. Figure 3.1 shows the architecture of the proposed methodology.

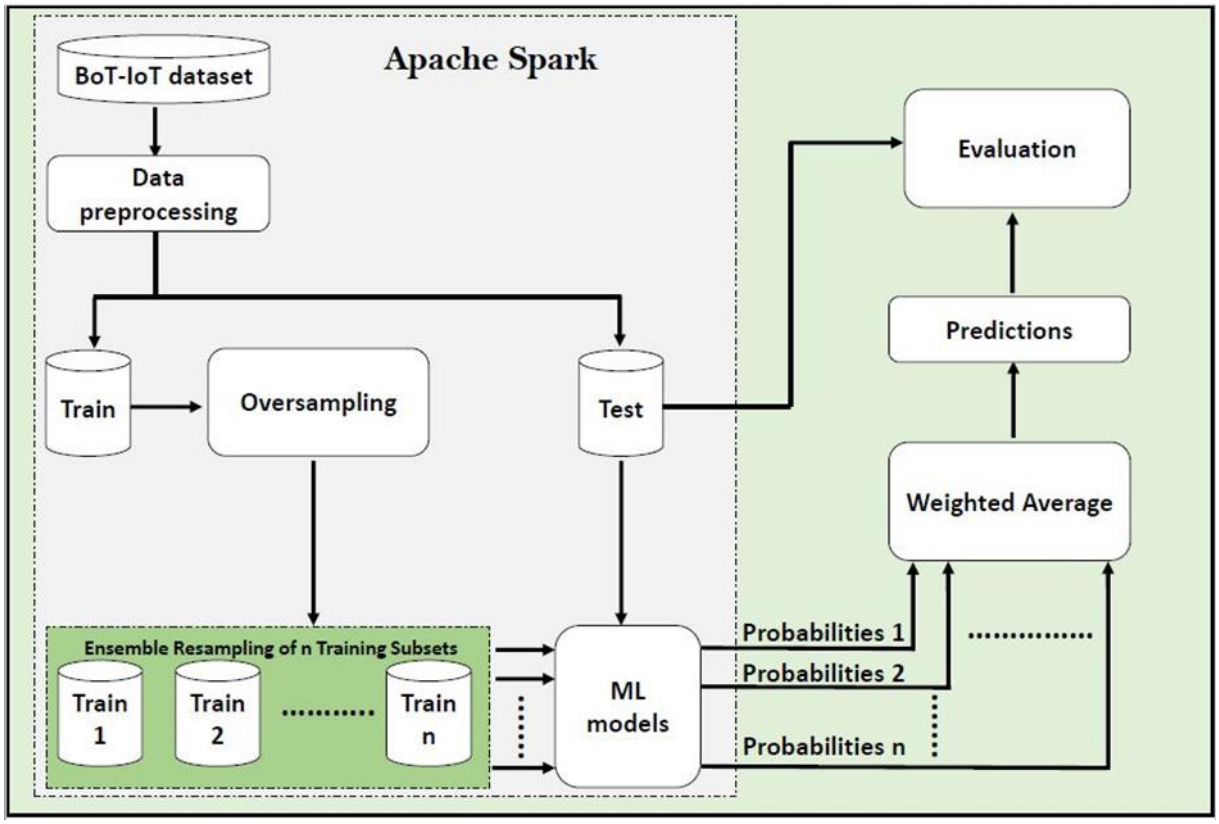


Figure 3. 1: Process of the proposed methodology.

3.3.2. Data Description

The creation of a realistic dataset is still a critical field of research when it comes to develop intrusion detection systems that detect cyberattacks. Numerous datasets have been

created in recent years, each with particular advantages and disadvantages. To successfully create a dataset of botnet traffic in IoT networks, Koroniotis et al [154] employed both real and simulated IoT network traffic. The dataset is known as Bot-IoT, and it includes a variety of attack types. The development of the Bot-IoT dataset involved the use of a realistic testbed, and a variety of tools to execute various botnet scenarios, as depicted in Figure 3.2. In order to guarantee the reliability of the dataset labeling procedure, a packet filtering firewall was employed.

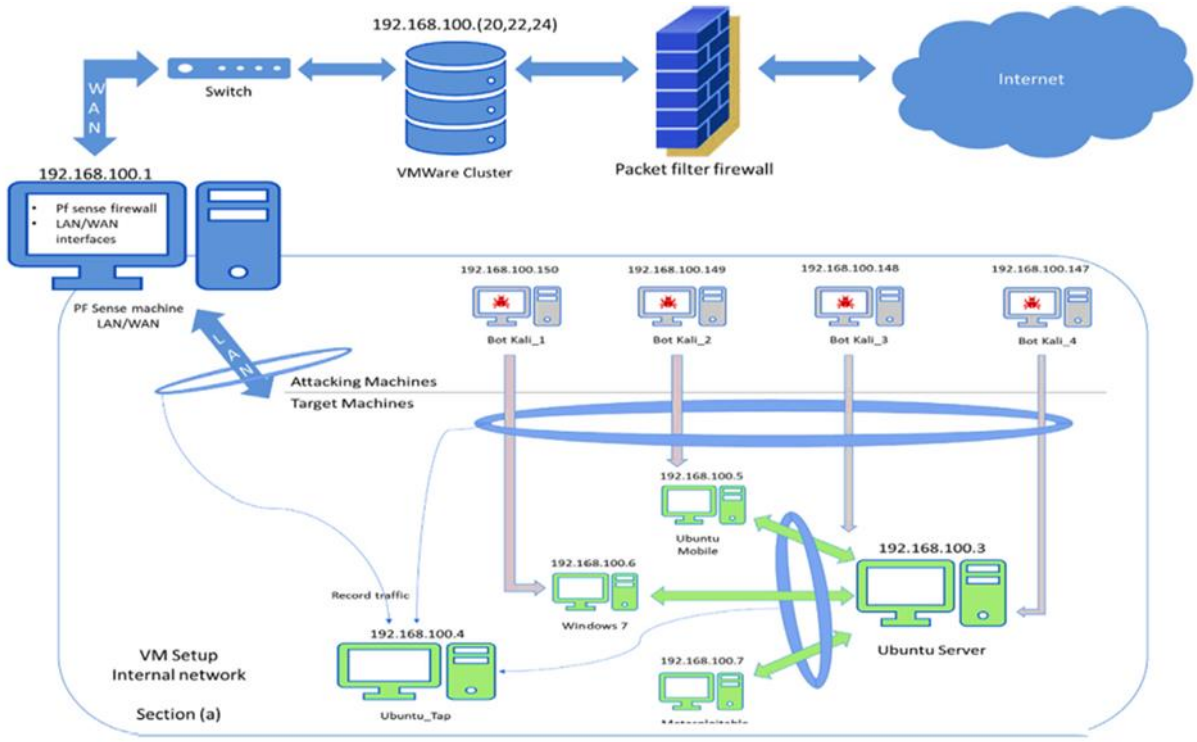


Figure 3. 2: The Bot-IoT dataset testbed environment [154].

Data features were generated by the Argus tool during the process of extracting network flow. These features were statistically examined via the joint entropy and correlation coefficient approaches. Given the huge quantity of data generated by IoT networks, big data analytics is necessary for the BoT-IoT dataset, which has over 73 million data samples. However, the BoT-IoT dataset is available in two formats: the short version and the full dataset. The datasets are extremely unbalanced, as Tables 3.3 and 3.4 demonstrate. Moreover, there are four attack types and one benign category. Each cyberattack type has a different collection of attack subcategories. The full BoT-IoT dataset was used in this study, which includes 32 network traffic features, as seen in Table

3.5. It is extremely unbalanced because the normal traffic and theft attack represent only 0.01% and 0.002% of the total data samples.

Table 3. 3: The full BoT-IoT dataset description.

Category	Subcategory	Number of records	Ratio (%)
Distributed Denial of Service (DDoS)	TCP	38532480	52.52
	UDP		
	HTTP		
Denial of Service (DoS)	TCP	33005194	44.98
	UDP		
	HTTP		
Reconnaissance	OS Fingerprinting	1821639	2.48
	Service Scanning		
Information Theft	Keylogging	1587	0.002
	Data Exfiltration		
Normal	Normal	9543	0.01

Table 3. 4: The short version BoT-IoT dataset description.

Category	Subcategory	Number of records	Ratio (%)
Distributed Denial of Service (DDoS)	TCP	1926624	52.52
	UDP		
	HTTP		
Denial of Service (DoS)	TCP	1650260	44.98
	UDP		
	HTTP		
Reconnaissance	OS Fingerprinting	91082	2.48
	Service Scanning		
Information Theft	Keylogging	79	0.002
	Data Exfiltration		
Normal	Normal	477	0.01

Table 3. 5: BoT-IoT dataset features description.

Features	Description
drate	Destination to source packets per second
srate	Source to destination packets per second
rate	Total packets per second in transaction
dbytes	Destination to source byte count
sbytes	Source to destination byte count
dpkts	Destination to source packet count
spkts	Source to destination packet count

Table 3.5 (continued)

max	Maximum duration of aggregated records
min	Minimum duration of aggregated records
sum	Total duration of aggregated records
stddev	Standard deviation of aggregated records
mean	Average duration of aggregated records
dur	Record total duration
seq	Argus sequence number
ltime	Record last time
state	Transaction state
bytes	Total number of bytes in transaction
pkts	Total count of packets in transaction
dport	Destination port number
daddr	Destination IP address
sport	Source port number
saddr	Source IP address
Proto	Textual representation of transaction protocols present in network flow
flgs	Flow state flags seen in transactions
Stime	Record start time
pkSeqID	Row Identifier

3.3.2. Data Preprocessing

3.3.2.1 Data Cleaning

To simplify and clean the input data, several data preprocessing procedures must be used after loading the dataset. Data cleaning was carried out in the first step of data preprocessing by eliminating features that contained null data and those that were invalid (see Table 3.6). Following the removal of undesired features, we eliminated any record that had missing values. In the next step, we converted all string values corresponding to the attack types and features "flgs", "proto" and "state" into integer values. Several times, the 'sport' and 'dport' features had hexadecimal values. Therefore, we transformed the values to 0 to indicate that their port value is invalid [155].

Table 3. 6: BoT-IoT features selection.

selected features			invalid features	null data
flgs	dur	dpkts	pkSeqID	smac
proto	mean	sbytes	saddr	dmac
sport	stddev	dbytes	ltime	soui
dport	sum	rate	stime	doui
pkts	min	srate	daddr	sco
bytes	max	drate	seq	dco
state	spkts			

3.3.2.2 Train-Test Split

The BoT-IoT dataset is extremely unbalanced, as was previously highlighted. Therefore, to ensure that there were a sufficient number of instances from Information Theft class and Normal class (minority classes) in train and test datasets, the dataset was manually split into train and test. This means that, as Table 3.7 explains, we select different proportions of data instances from each class.

Table 3. 7: Distribution of instances in the training and testing datasets.

Categories	Train data	Test data
Normal	7314 (80.88%)	1758 (19.12%)
Theft	1029 (65.34%)	549 (34.66%)
Reconnaissance	1532782 (84.16%)	288409 (15.84%)
DoS	32150808 (97.42%)	852288 (2.58%)
DDoS	37342475 (96.92%)	1186516 (3.08%)
All	71034408 (96.82%)	2329520 (3.18%)

3.3.2.3 Handling Imbalanced Class Distribution

In general, when an unbalanced dataset is classified, the majority classes are favored, which leads to the problem of misclassifying the minority classes. In order to improve the accuracy of predictions of the unbalanced classification, we employed a combination of oversampling and ensemble resampling in the aim to reduce the bias between classes. The training dataset is used by the ensemble resampling approach to generate n training subsets from the majority classes (DoS, DDoS), which are then concatenated with the minority classes (Normal, Theft, and Reconnaissance). Figure 3.3 illustrates how all attack

subcategories are included on each of the n training subsets and the test set, despite the fact that our work is limited to classifying only the attack classes.

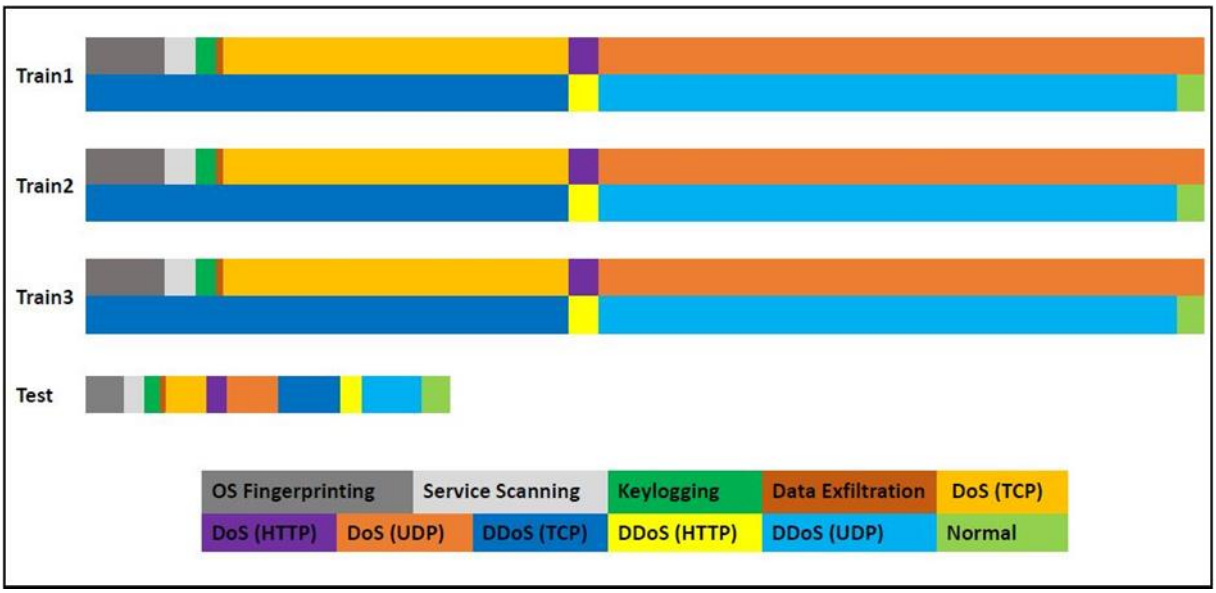


Figure 3. 3: Attack subcategory distribution in testing dataset and training subsets for n = 3.

From the Train data, we replicated 7314 data instances from the Normal category to produce 300,000 extra data instances during the oversampling approach. In a similar way, we generated 200,000 additional data instances by replicating 1029 data instances taken from the Information Theft class. Subsequently, we combined each of the n training subsets with the newly generated data instances. Tables 3.8 and 3.9 show the new training subset distribution per class category. Finally, we compared the results of the SMOTE oversampling approach and the data duplication approach. Through the following process, SMOTE creates artificial samples for the minority class [127]:

- The number of synthetic instances for each minority class is determined.
- A random instance of the minority class is chosen.
- The KNN algorithm is utilized to determine the K nearest neighbors of the chosen instance.
- One of the K instances is chosen at random.
- Through random interpolation, a new synthetic instance is created from the minority class instance and the chosen neighbor minority class instance.

- The operations in the previous four steps are repeated until the required amount of synthetic instances is achieved.

Table 3. 8: Distribution of instances in the training subsets when n=3.

Categories	Train 1	Train 2	Train 3
DDoS	12676809	12676720	12676683
DoS	12153172	12153335	12153367
Reconnaissance	1532782	1532782	1532782
Theft	201029	201029	201029
Normal	305881	305748	305687
All	26869673	26869614	26869548

Table 3. 9: Distribution of instances in the training subsets when n=6.

Categories	Train 1	Train 2	Train 3	Train 4	Train 5	Train 6
Normal	305462	305385	305357	305361	305334	305317
Theft	201029	201029	201029	201029	201029	201029
Reconnaissance	1532782	1532782	1532782	1532782	1532782	1532782
DoS	6483181	6483984	6483028	6483251	6483612	6483823
DDoS	6677066	6677468	6677435	6676859	6676470	6677343
All	15199520	15200648	15199631	15199282	15199227	15200294

3.3.3. Classification

Predicting whether the flow of network traffic is malicious or benign is the objective of the classification process. For big data classification, Spark machine learning (ML) offers a set of tools and algorithms for effectively completing the classification task at scale, by utilizing Apache Spark's distributed computing capabilities.

4.6.2.1 Apache Spark

Apache Spark is the most commonly utilized framework for managing big data applications. The main advantage of Spark is its in-memory processing, which enables swift data processing. Additionally, it has proven to be incredibly scalable, maintaining performance even when several nodes are active. Spark works at a rate much faster than datasets working on hard drives because it is built on the Resilient Distributed Dataset (RDD), which is able to be kept in memory on cluster working nodes and is separated into various partitions [156]. For every program, a slave process named an executor is

established in each worker node. Its job is to run the tasks and cache the data in memory or drive. Task scheduling is performed by a master process known as driver that is created by each Spark program [157]. In order to run their Spark programs, Different cluster modes enable driver processes to establish connections with their standalone cluster manager or other popular cluster managers such as YARN [158], Mesos [159]. A diagram of Spark's cluster architecture is shown in Figure 3.4.

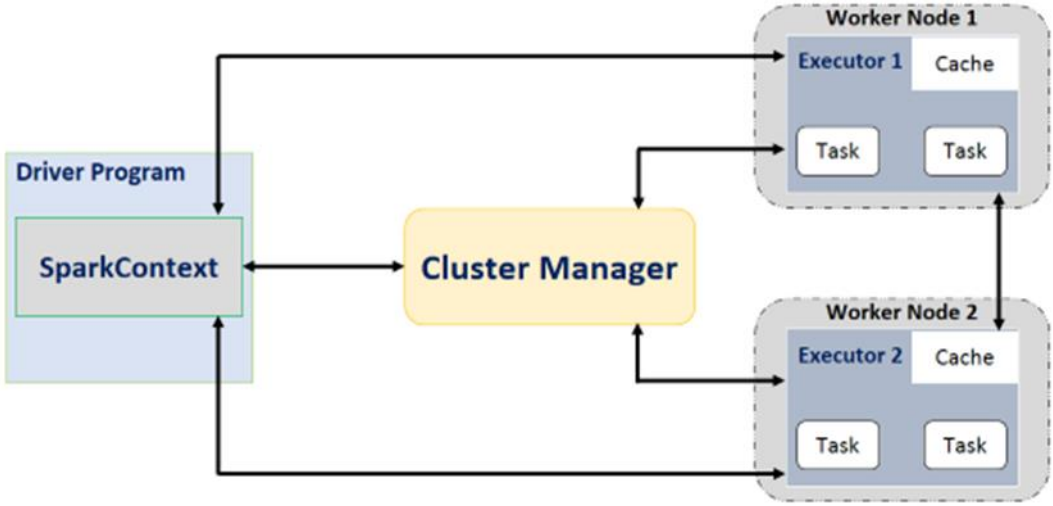


Figure 3. 4: The spark cluster architecture

3.3.3.2 Apache Spark MLlib

As one of the most popular open-source libraries for big-data ML, Apache Spark MLlib provides classification algorithms that make writing implementations considerably easier. It includes a variety of ML tools for performing various tasks like: managing data, saving and load models, tuning ML pipelines, feature selection, feature extraction, etc. Several algorithms for multiclass and binary classification are available in MLlib [160]. The following is a list of supported algorithms for both scenarios.

Multiclass Classification: Naive Bayes, Random Forests, Decision Trees, Logistic Regression.

Binary Classification: Naive Bayes, Gradient-Boosted Trees, Random Forests, Decision Trees, Logistic Regression, linear SVMs.

3.3.3.3 Classifier

Machine learning ML algorithms exhibit a bias towards the majority classes when performing an unbalanced Big Data classification task. In the full BoT-IoT dataset, the decision tree (DT) provided superior classification results than other Apache Spark ML algorithms [151]. On the other hand, Spark is obviously unsuitable for DL because algorithms such as RNN, CNN, and ANN are computationally expensive and challenging to integrate with Apache Spark. As a result, we used a decision tree, which is like most ML algorithms, is vulnerable to the imbalanced problem. In this work, the results of DT models that had been trained on n training subsets were combined using a weighted average ensemble. The main idea is that every training subset produces a different result from the DT algorithm. As a consequence, the model's outputs balance each other out when combined. In the end, we utilized the averaging ensemble method to compare the weighted average results.

3.4 Conclusion

This chapter introduced a new big data architecture based on Apache Spark for detecting IoT network intrusions. To achieve this, oversampling and ensemble resampling techniques were combined in the first step to solve the issue of class imbalance. After that, we built a number of DT models for multiclass classification. In the end, the final predictions were obtained by applying a weighted average ensemble. After converting the Bot-IoT dataset into a suitable format during the data preprocessing phase, we tested the effectiveness of our methodology.

CHAPTER 4: EXPERIMENTAL EVALUATION AND RESULTS

4.1 Introduction

The experiments carried out to analyze Internet of Things data using artificial intelligence (AI) technologies are covered in Chapter 4. Our approach involves monitoring road surface conditions utilizing the RSC and RSC-IoT datasets. Further, we utilized the BoT-IoT dataset to detect IoT network intrusions. Through RSC and RSC-IoT datasets, hybrid deep learning models were trained to detect road surface anomalies in a multi-class classification problem with three classes. Likewise, the BoT-IoT dataset was used to train ML models in Apache Spark for detecting IoT network intrusions in a five-class classification situation. Google Colaboratory [161] was utilized to execute our methodology and evaluate the models that we created. All AI algorithms were written in Python, and graphical plots and performance metrics have been employed to evaluate the results. Based on the comparisons, the proposed algorithm's efficiency is verified.

4.2 Performance Metrics

Since our datasets suffer from class imbalance, some metrics can be misleading [162]. For this, we utilized metrics, namely Precision, Sensitivity (or Recall), Specificity, F1-score, and Geometric Mean to assess the classification performance. These metrics treat majority classes and minority classes equally, which means that poor performance of minority classes will reduce the overall performance [163]. As demonstrated in Figure 4.1 and Table 4.1, the metrics are based on the confusion matrix.

A confusion matrix has become frequently utilized to visualize the reliability of AI classification algorithms. A confusion matrix is a table that offers an extensive look at the classification results and is composed of the following four components [164]:

- True Positive (TP): A count of positive events accurately predicted to be positive by the AI algorithm.
- True Negative (TN): A count of negative events accurately predicted to be negative by the AI algorithm.
- False Positive (FP): A count of negative events wrongly predicted to be positive by the AI algorithm.

- False Negative (FN): A count of positive events wrongly predicted to be negative by the AI algorithm.

Since errors have unique effects, it is crucial to distinguish between false positives and negatives. Usually, columns display the predicted classes and rows display the actual classes.

	Positive Prediction	Negative Prediction
Positive Class	True Positive (TP)	False Negative (FN)
Negative Class	False Positive (FP)	True Negative (TN)

Figure 4. 1: The confusion matrix's structure

Table 4. 1: Performance metrics based on confusion matrix.

Metrics	Formula
Precision	$\frac{TP}{TP + FP}$
Recall (Sensitivity)	$\frac{TP}{TP + FN}$
Specificity	$\frac{TN}{TN + FP}$
F1-score	$\frac{2 \times Sensitivity \times Precision}{Sensitivity + Precision}$
G-Mean	$\sqrt{Sensitivity \times \text{Specificity}}$

4.3 RSC Monitoring Experiments using the RSC Dataset

As mentioned earlier, several hybrid deep learning models were developed for road surface anomaly classification. The RSC dataset was utilized in the first and third proposed frameworks that were described in Chapter 2. This section covers the experiments carried out in the first proposed framework [165]. Additionally, we randomly assigned 70% of our dataset to be utilized for training and the rest 30% to test purposes for each experiment.

4.3.1. Performance for Different Input Domains and Data Types

This part covers the experiments that were carried out to determine the most suitable combination of sensors and input features. By utilizing CNN and DNN models with 50%

overlap, we examine the effects of the input data that was used for monitoring road surface conditions. This means that selecting the appropriate type of sensor has a considerable impact on road anomaly detection ability. Three different sensor combinations were used to classify the road data, as shown in Table 4.2. The abbreviations Azimuth, Pitch, Roll, Gx, Gy, Gz, Ax, Ay, and Az corresponded to the orientation angles, three-axis gyroscope, and three-axis accelerometer, respectively.

Table 4. 2: Performance for different sensor combinations using DNN and CNN models.

Input	Model	Shape	F1 Score	Recall	Precision	Accuracy
Ax, Ay, Az	DNN	(None,150)	0.7519	0.7146	0.8075	0.9012
	CNN	(None,1,50,3)	0.8068	0.7814	0.8445	0.9186
Ax, Ay, Az, Gx, Gy, Gz	DNN	(None,300)	0.7863	0.7460	0.8426	0.9186
	CNN	(None,1,50,6)	0.8504	0.8130	0.8980	0.9406
Ax, Ay, Az, Gx, Gy, Gz, Azimuth, Pitch, Roll	DNN	(None,450)	0.8243	0.7991	0.8539	0.9259
	CNN	(None,1,50,9)	0.8678	0.8515	0.8857	0.9459

In fact, the F1 score, recall, precision, and accuracy were used for evaluating these sensors' performance. Orientation, gyroscope, and Accelerometer are used in combination to provide DNN and CNN classifiers with 92% and 94% accuracy, respectively. This is a 2% and 3% increase in accuracy over the employment of the accelerometer individually. Furthermore, when compared to the gyroscope and accelerometer combination, which produced 74%, 81% recall and 78%, 85% F1 score, the orientation, gyroscope, and accelerometer combination achieved much higher results with 79%, 85% recall, and 82%, 86% F1 score. The 4% recall increase demonstrates the importance of employing orientation angles for enhancing the efficiency of recognition, particularly when distinguishing between man-made and real road anomalies.

The study presented here additionally examines the results of transforming the original sensor signals (input data) from the time domain into the frequency domain and wavelet transformation. Table 4.3 below shows the results of multiple experiments using FFT amplitude components and various wavelet families, like Daubechies 6 (Db 6), Daubechies 10 (Db10), Symlets 5, Haar, and Reverse Biorthogonal 3.1. Additionally, when using the Haar wavelets and FFT amplitude components, the CNN F1 score and accuracy are 87% and 95%, respectively, whereas for all other wavelet families, the results are F1 score of

86% and 94% accuracy. The table makes it clear that, in comparison to the time-domain input, the FFT-DWT combination offers better classification results.

Table 4. 3: Performance for multiple features using DNN and CNN models.

Input	Model	Shape	F1 Score	Recall	Precision	Accuracy
FFT (amplitude)	(None,450)	DNN	0.8563	0.8408	0.8738	0.9453
	(None,1,50,9)	CNN	0.8765	0.8608	0.8936	0.9519
Db10	(None,612)	DNN	0.7542	0.7235	0.7933	0.8979
	(None,1,68,9)	CNN	0.8610	0.8340	0.8965	0.9419
Db6	(None,540)	DNN	0.7751	0.7298	0.8388	0.9085
	(None,1,60,9)	CNN	0.8682	0.8503	0.8878	0.9459
Sym5	(None,522)	DNN	0.7590	0.7192	0.8272	0.9052
	(None,1,58,9)	CNN	0.8634	0.8346	0.8975	0.9453
haar	(None,450)	DNN	0.7934	0.7735	0.8162	0.9166
	(None,1,50,9)	CNN	0.8775	0.8561	0.9015	0.9526
bior3.1	(None,468)	DNN	0.7758	0.7360	0.8298	0.9052
	(None,1,52,9)	CNN	0.8686	0.8432	0.8990	0.9439
FFT(amplitude)+haar	(None,900)	DNN	0.8689	0.8548	0.8868	0.9466
	(None,900)	CNN	0.8805	0.8655	0.8968	0.9546

4.3.2. A Comparison of Hybrid Classification Models

Following the implementation of the proposed models, we discovered that both the CNN-LSTM and CNN-GRU models performed better than the CNN model. The hybrid model's performance was improved as a result of GRU and LSTM's ability to capture certain feature dependencies [166].

With three different overlapping factors, Table 4.4 displays the classification performance results. The results clearly demonstrated that the best performance was achieved with a 66% overlap. Achieving an F1 score of 92.41% and an accuracy of 97.06%, the CNN-GRU model performed slightly better than the CNN-LSTM model, which achieved a 91.37% F1 score and a 96.67% accuracy.

Figures 4.2, 4.3, and 4.4 display the confusion matrices for the classification results of the three models, in which we employed overlaps of 33%, 50%, and 66%. According to the confusion matrices, we can see that the majority of the smooth road segments were accurately identified. Furthermore, it seems that the models experienced several challenges when it came to distinguishing between man-made road anomalies like speed bump and

speed hump and real road anomalies like pothole and crack. On the other hand, compared to the CNN-LSTM model, the CNN-GRU model achieves better results, this validates earlier results in the fields of human activity recognition [99] and electric energy forecasting [102].

Table 4. 4: Evaluation of the presented models using multiple overlap factors.

overlap	Model	F1 Score	Recall	Precision	Accuracy
33%	CNN-GRU	0.8857	0.8763	0.8972	0.9566
	CNN-LSTM	0.8888	0.8794	0.9002	0.9566
	CNN	0.8772	0.8625	0.8933	0.9520
50%	CNN-GRU	0.8879	0.8833	0.8935	0.9579
	CNN-LSTM	0.8920	0.8816	0.9030	0.9586
	CNN	0.8805	0.8655	0.8968	0.9546
66%	CNN-GRU	0.9241	0.9167	0.9318	0.9706
	CNN-LSTM	0.9137	0.9048	0.9230	0.9667
	CNN	0.9077	0.8977	0.9181	0.9620

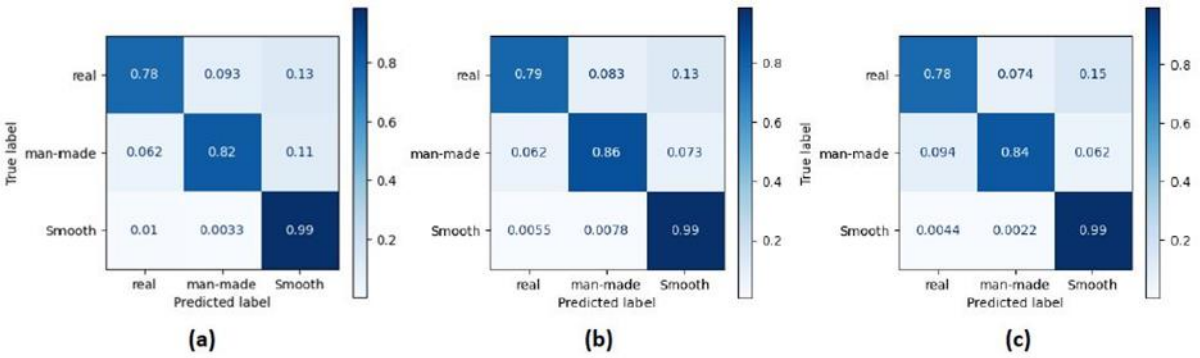


Figure 4. 2: Normalized confusion matrices of 33% overlaps for the (a) CNN, (b) CNN-LSTM, (c) and CNN-GRU classifiers.

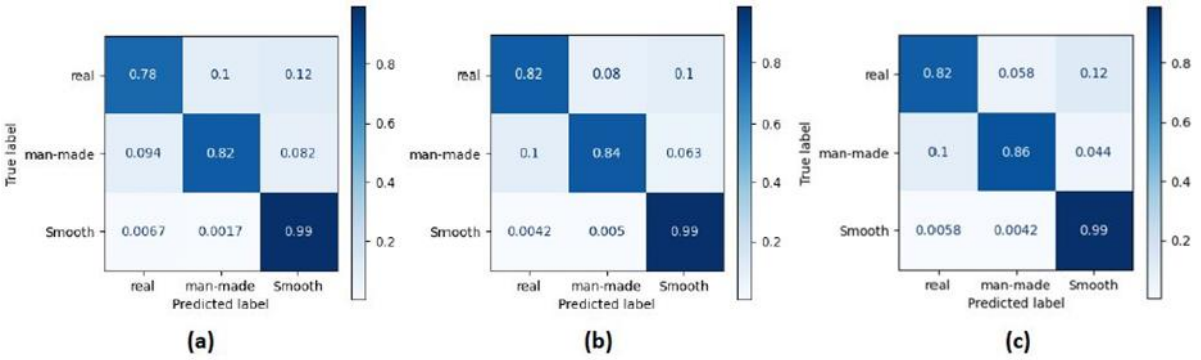


Figure 4. 3: Normalized confusion matrices of 50% overlaps for the (a) CNN, (b) CNN-LSTM, (c) and CNN-GRU classifiers.

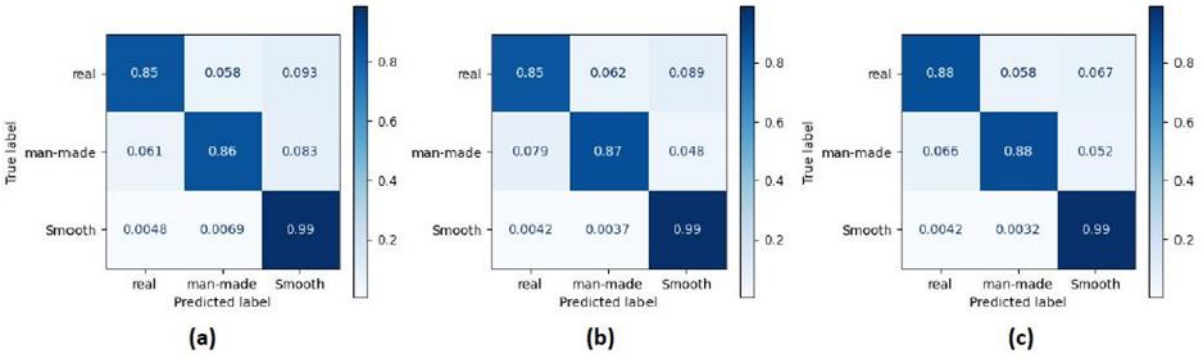


Figure 4. 4: Normalized confusion matrices of 66% overlaps for the (a) CNN, (b) CNN-LSTM, (c) and CNN-GRU classifiers.

4.4 RSC Monitoring Experiments using the RSC-IoT Dataset

This section presents the performance evaluation of the experiments carried out for RSC monitoring utilizing the RSC-IoT dataset. A comparative analysis is conducted between the standard hybrid deep learning models and the 3D hybrid deep learning models that have been presented in the second proposed framework (Chapter 2). Table 4.5 and Figures 4.5, 4.6, 4.7, 4.8, and 4.9, show that TD-CNN-GRU and TD-CNN-LSTM performed better than the standard CNN-GRU and CNN-LSTM, whereas the ConvLSTM outperformed the standard models slightly. The results of the prediction highlight the benefits of using the TimeDistributed layer for analyzing time series data in RSC monitoring. In addition, the advantage of ConvLSTM over CNN-LSTM demonstrates that convolution operation, as an alternative to matrix multiplication [167, 168], produces better results in RSC monitoring.

Table 4. 5: A comparative evaluation of the proposed models.

Method	Input shape	F1 Score	Recall	Precision	Accuracy
ConvLSTM	(4,10,6,1)	0.8998	0.8611	0.9475	0.9449
TD-CNN-GRU	(4,10,6,1)	0.9202	0.9002	0.9427	0.9535
TD-CNN-LSTM	(4,10,6,1)	0.9300	0.9201	0.9404	0.9573
CNN-GRU	(10,6,4)	0.8933	0.8556	0.9405	0.9410
CNN-LSTM	(10,6,4)	0.8887	0.8536	0.9318	0.9426

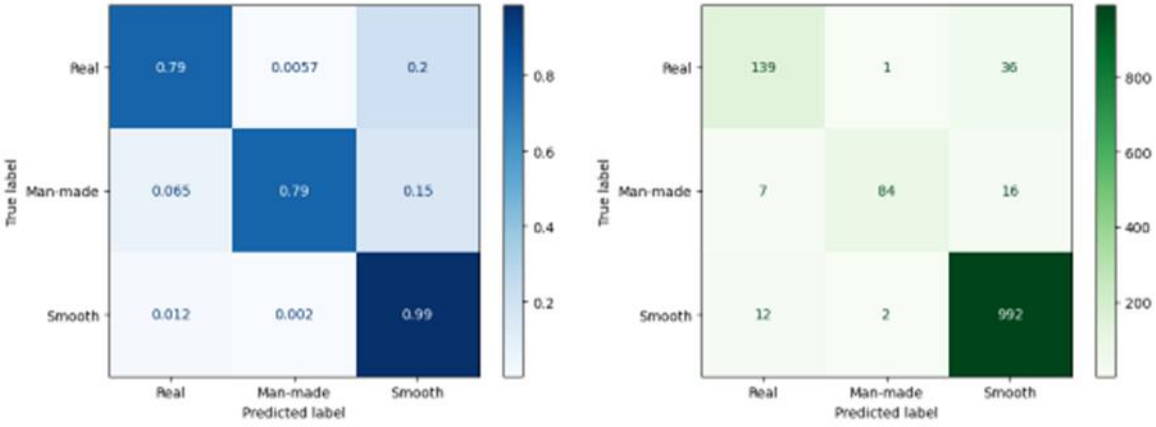


Figure 4. 5: Confusion matrices of the standard CNN-LSTM.

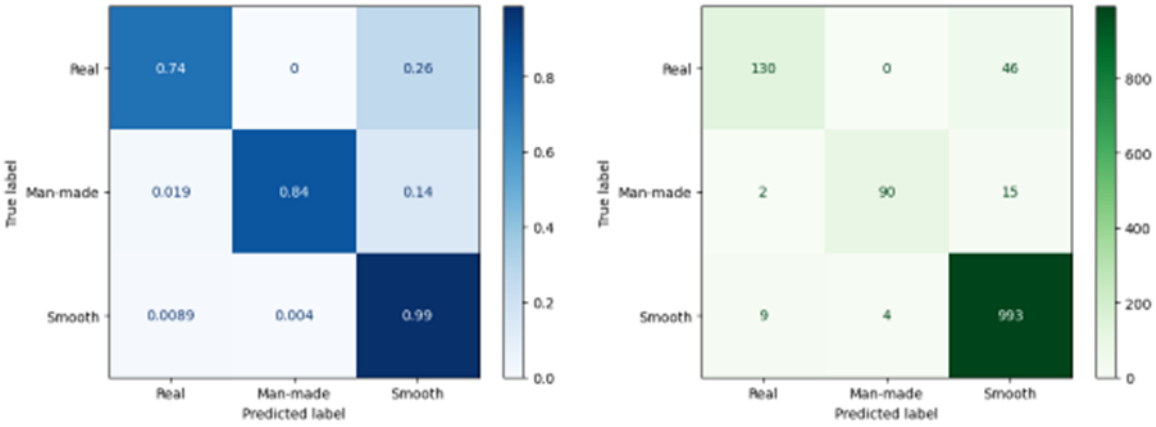


Figure 4. 6: Confusion matrices of the standard CNN-GRU.

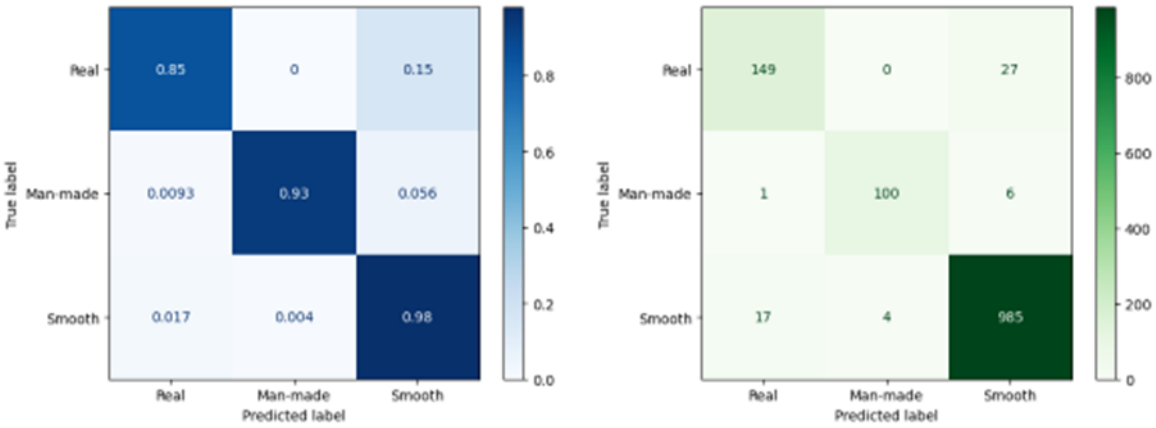


Figure 4. 7: Confusion matrices of the TD-CNN-LSTM.

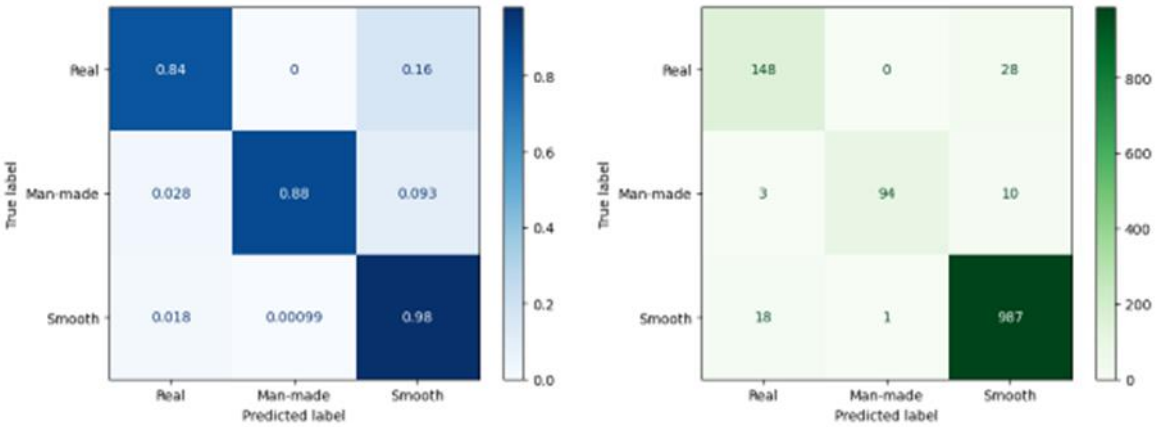


Figure 4. 8: Confusion matrices of the TD-CNN-GRU.

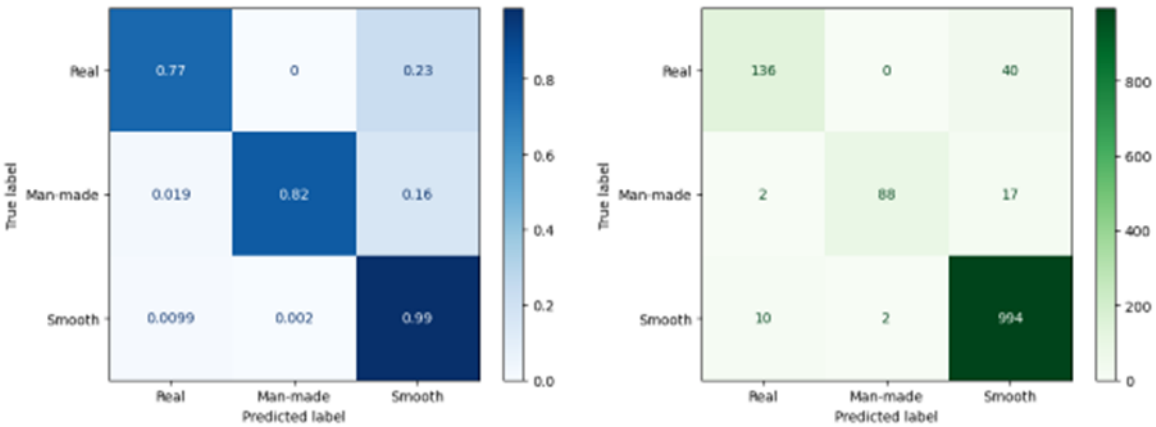


Figure 4. 9: Confusion matrices of the ConvLSTM.

In order to lower the variance in the prediction errors, the predicted results obtained from TD-CNN-LSTM, TD-CNN-GRU, and ConvLSTM have been combined to build a weighted average ensemble and an averaging ensemble. A comparison of the weighted average ensemble with the averaging ensemble is presented in Table 4.6 and Figures 4.10, and 4.11. The results indicate that the weighted average ensemble outperformed the averaging ensemble. This is because the performance of the averaging ensemble was negatively impacted by the ConvLSTM model's inferiority to the TD models. When the weighted average ensemble is used, the weights of the ConvLSTM, TD-CNN-GRU, and TD-CNN-LSTM are obtained as follows: $w_1 = 0.1$, $w_2 = 0.1$, and $w_3 = 0.2$. Because we are dealing with an imbalanced dataset, we employed grid search to find the most effective combination of w_1, w_2, w_3 that results in the highest F1-score, which is the right metric for imbalanced data.

Table 4. 6: Performance evaluation of the ensemble methods.

Method	F1 Score	Recall	Precision	Accuracy
Averaging	0.9293	0.9049	0.9567	0.9589
Weighted Average	0.9335	0.9143	0.9546	0.9604

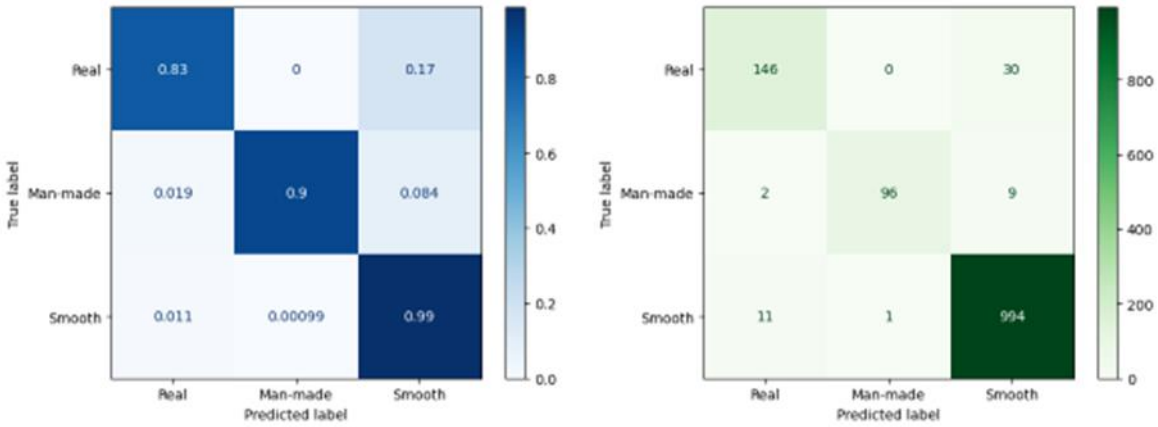


Figure 4. 10: Confusion matrices of the averaging ensemble.

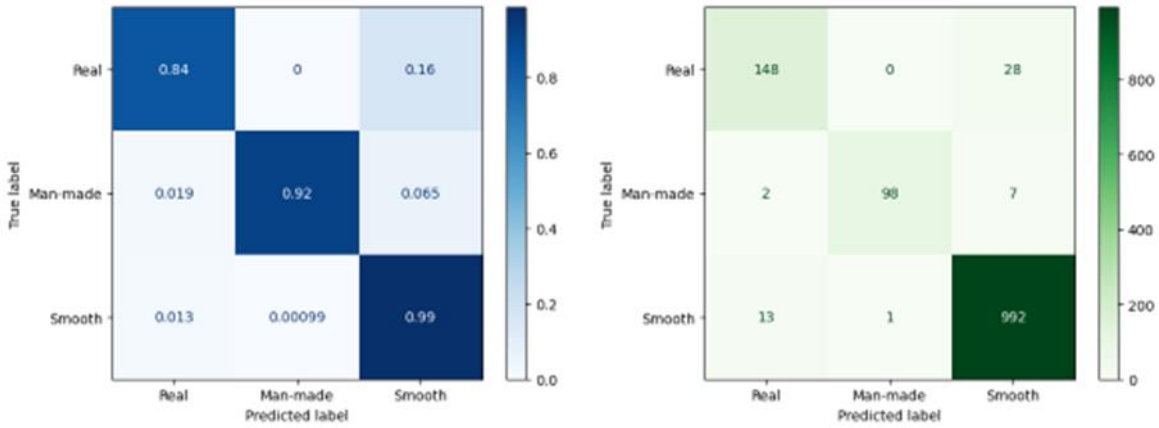


Figure 4. 11: Confusion matrices of the weighted average ensemble.

4.5 RSC Monitoring Experiments using Data Augmentation

The performance of the experiments conducted for the third proposed framework that was presented in Chapter 2 is shown in this section. We selected at random 70% of the RSC dataset to be used for training and the remaining 30% for testing. The results of each evaluated model for classifying road surfaces are displayed in Table 4.7. As we can see, the 3D models generated the best results. The 3D models significantly outperformed both of the standard hybrid deep learning models. Furthermore, the TD-CNN-GRU model achieved the best performance with an F1 score of 94.86%. In contrast to the second proposed framework, the ConvLSTM model had the lowest performance, with an F1 score of 87.15 %.

Table 4. 7: Performance evaluation of the proposed models without data augmentation.

Model	Input shape	F1 Score	Recall	Precision	Accuracy
CNN-LSTM	(25,18,3)	0.8972	0.8712	0.9306	0.9791
CNN-GRU	(25,18,3)	0.8806	0.8328	0.9440	0.9747
TD-CNN-LSTM	(3,25,18,1)	0.9248	0.9091	0.9417	0.9866
TD-CNN-GRU	(3,25,18,1)	0.9486	0.9394	0.9603	0.9896
ConvLSTM	(3,25,18,1)	0.8715	0.8409	0.9132	0.9747

In order to determine the effect of data augmentation on the model's performance, three different data augmentation techniques have been proposed for the training dataset, as indicated in Table 4.8. It is important to remember that the test data used for evaluating the 3D hybrid deep learning models had been separated before applying data augmentation.

This makes comparing the 3D models and evaluating the effect of data augmentation easier.

Table 4.8 makes it clear that adding augmented data to the set of training data improves the performance of the 3D models. To determine the most effective data augmentation technique, we looked into the averaging ensemble of TD-CNN-LSTM, TD-CNN-GRU, and ConvLSTM, which combines the results obtained from these 3D models. As we can see, the DoppelGANger technique achieved the highest F1 Score, Recall, and Accuracy at 96.20%, 96.21%, and 99.40%, respectively. Whereas the SMOTE technique achieved the highest precision of 96.75%. Figures 4.12, 4.13, and 4.14 depict the confusion matrices of the averaging ensemble for each data augmentation technique. In these figures, we can see that all smooth road segments were correctly classified, while only a small percentage of man-made and real road anomalies misclassified.

Table 4. 8: Performance evaluation of the proposed models with data augmentation.

Model	Input shape	F1 Score	Recall	Precision	Accuracy
Traditional approaches					
TD-CNN-LSTM	(3,25,18,1)	0.9475	0.9313	0.9705	0.9911
TD-CNN-GRU	(3,25,18,1)	0.9472	0.9540	0.9419	0.9911
ConvLSTM	(3,25,18,1)	0.9045	0.8923	0.9201	0.9791
Averaging	(3,25,18,1)	0.9493	0.9394	0.9599	0.9911
SMOTE					
TD-CNN-LSTM	(3,25,18,1)	0.9376	0.9313	0.9442	0.9881
TD-CNN-GRU	(3,25,18,1)	0.9613	0.9545	0.9684	0.9940
ConvLSTM	(3,25,18,1)	0.8985	0.8561	0.9497	0.9791
Averaging	(3,25,18,1)	0.9570	0.9470	0.9675	0.9925
DoppelGANger					
TD-CNN-LSTM	(3,25,18,1)	0.9644	0.9545	0.9751	0.9925
TD-CNN-GRU	(3,25,18,1)	0.9510	0.9610	0.9423	0.9911
ConvLSTM	(3,25,18,1)	0.8795	0.8625	0.8992	0.9791
Averaging	(3,25,18,1)	0.9620	0.9621	0.9625	0.9940

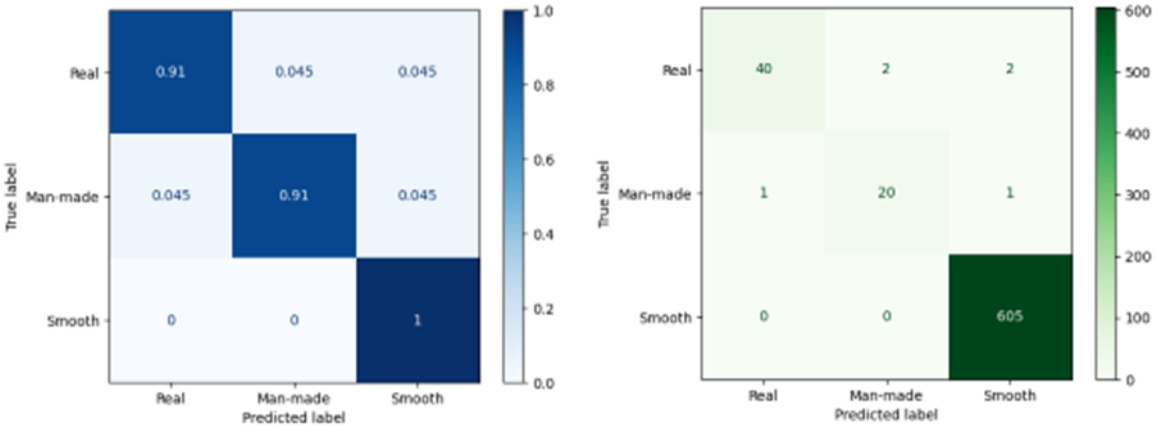


Figure 4. 12: Confusion matrices of the averaging ensemble using Traditional approaches.

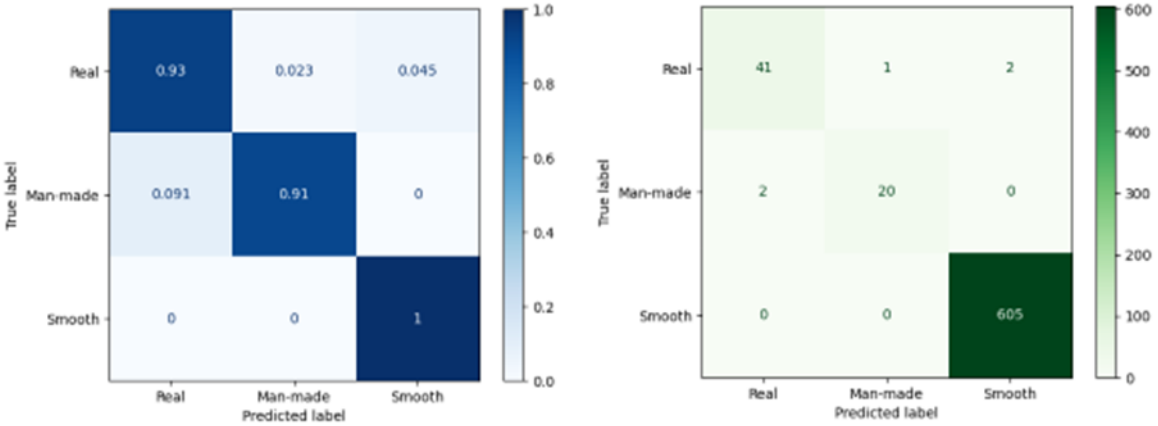


Figure 4. 13: Confusion matrices of the averaging ensemble using SMOTE technique.

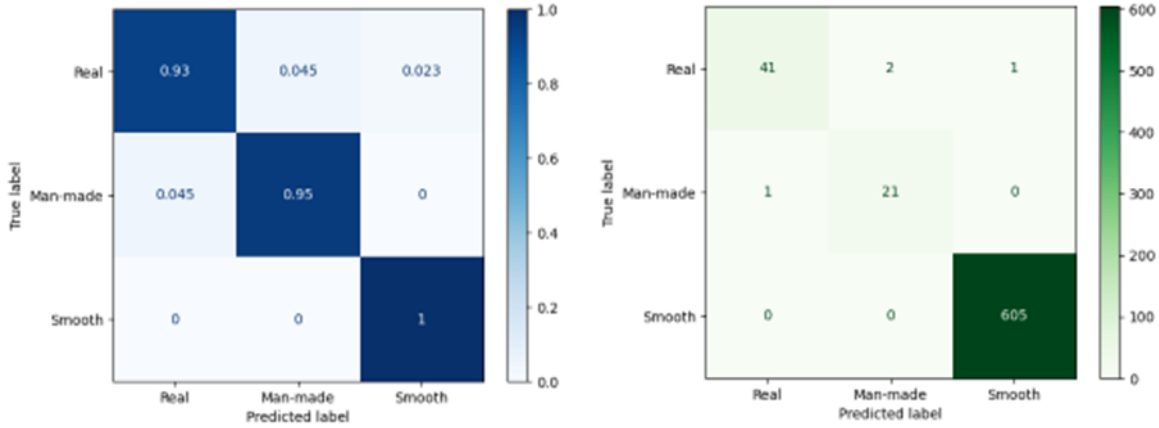


Figure 4. 14: Confusion matrices of the averaging ensemble using DoppelGANger technique.

4.6 IoT Intrusion Detection Experiments using the BoT-IoT Dataset

The performance analysis of the intrusion detection experiments using the BoT-IoT dataset is presented in this section. The experiments were conducted on Google Colaboratory utilizing the Apache Spark environment, which minimizes memory use and keeps the system from becoming overloaded.

4.6.2. Importance of using Oversampling on the BoT-IoT Dataset

The impact of oversampling techniques on the BoT-IoT dataset is demonstrated by the classification results seen in Table 4.9. We classified attack categories using the DT algorithm during the big data classification phase. As anticipated, in the majority of evaluation metrics, the experimental results with the resampled datasets performed better than those obtained with the non-resampled datasets. Additionally, two stages of investigation were done on the impact of oversampling. In the beginning, data instances from the Theft attack class and the Normal network traffic class were duplicated to resample the training dataset. After that, we utilized the data replication technique to duplicate data instances from the Theft attack class and SMOTE to create new synthetic data from the Normal class to resample the training dataset. The experiment demonstrated that using the SMOTE technique in conjunction with data duplication resulted in lower classification results than using data duplication individually. This is a result of SMOTE producing artificial data without taking neighboring instances of other classes into account. Large datasets typically result in less successful outcomes from SMOTE, which increases noise and classes overlapping [163]. Even so, just 45% of DoS attack class were properly recognized, whereas the most accurate results identified 12% of the normal class as attacks. Therefore, in order to enhance the classification performance, more sophisticated techniques must be used.

Table 4. 9: Classification results using multiple oversampling techniques on the BoT-IoT dataset.

	Precision	Sensitivity	specificity	F1 Score	G-Mean
no resampling					
DDoS	0.85	0.81	0.85	0.83	0.83
DoS	0.90	0.81	0.95	0.85	0.88
Reconnaissance	0.65	0.97	0.93	0.78	0.95
Theft	0.00	0.00	1.00	0.00	0.00
Normal	1.00	0.02	1.00	0.04	0.15
macro avg	0.68	0.52	0.95	0.50	0.70
resampling with data duplication					
DDoS	0.71	0.95	0.59	0.81	0.75
DoS	0.88	0.45	0.96	0.60	0.66
Reconnaissance	0.99	0.95	1.00	0.97	0.98
Theft	0.12	0.91	1.00	0.21	0.96
Normal	0.20	0.88	1.00	0.33	0.94
macro avg	0.58	0.83	0.91	0.58	0.87
resampling with smote and data duplication					
DDoS	0.83	0.87	0.81	0.85	0.84
DoS	0.81	0.76	0.90	0.78	0.82
Reconnaissance	1.00	0.95	1.00	0.97	0.97
Theft	0.07	0.92	1.00	0.13	0.96
Normal	0.50	0.08	1.00	0.13	0.28
macro avg	0.64	0.71	0.94	0.57	0.82

4.6.2. Performance Evaluation of Ensemble Learning Methods

4.6.2.1 Averaging Ensemble

The approach we propose to detect intrusions depends on an ensemble resampling technique and uses DT models. The training dataset was split into n separate training subsets at the beginning of the studies, and these subsets were then resampled via data duplication technique. After that, DT models were trained on the Apache Spark environment using the training subsets. At last, the final predictions have been determined by taking the average of the models' class probability predictions. Tables 4.10 and 4.11 show the results obtained by all DT models and the averaging ensemble method with $n = 3$ and $n = 6$, respectively. The performance metrics show that the results produced by averaging ensemble with $n = 3$ outperformed the results obtained with n = 6. Unlike the case of n =3, some individual models performed better than the averaging ensemble of n =6. This was because some models in the ensemble like model 1 and model 2 were

very bad which negatively affected the performance of the averaging ensemble. In the case of n =3 the classification results obtained using the averaging ensemble were good for all the class categories except the Theft attack class.

Figures 4.15 and 4.16 display the confusion matrices of averaging ensemble when $n =$ 3 and $n = 6$. As can be shown, the minority classes for $n = 3$ and $n = 6$ got excellent Sensitivity scores, with the exception of the Theft attack class at $n = 6$. Therefore, we can say that the imbalance problem affects more on Theft attack class.

Table 4. 10: Performance comparison of the three DT models and the averaging ensemble method

	Precision	Sensitivity	specificity	F1 Score	G-Mean
model 1	0.62	0.66	0.92	0.62	0.78
model 2	0.61	0.85	0.93	0.62	0.89
model 3	0.54	0.81	0.88	0.59	0.85
Averaging ensemble					
DDoS	0.86	0.83	0.86	0.85	0.85
DoS	0.78	0.83	0.86	0.81	0.85
Reconnaissance	1.00	0.94	1.00	0.97	0.97
Theft	0.33	0.91	1.00	0.49	0.95
Normal	0.79	0.97	1.00	0.87	0.99
macro avg	0.75	0.90	0.95	0.80	0.92

Table 4. 11: Performance comparison of the six DT models and the averaging ensemble method

	Precision	Sensitivity	specificity	F1 Score	G-Mean
model 1	0.56	0.72	0.95	0.57	0.83
model 2	0.53	0.84	0.92	0.54	0.88
model 3	0.56	0.63	0.94	0.52	0.77
model 4	0.66	0.76	0.87	0.68	0.81
model 5	0.74	0.63	0.87	0.60	0.74
model 6	0.67	0.82	0.88	0.69	0.85
Averaging ensemble					
DDoS	0.82	0.75	0.82	0.78	0.79
DoS	0.69	0.76	0.80	0.72	0.78
Reconnaissance	1.00	0.99	1.00	0.99	0.99
Theft	0.54	0.10	1.00	0.17	0.32
Normal	0.25	0.97	1.00	0.40	0.99
macro avg	0.66	0.71	0.92	0.61	0.81

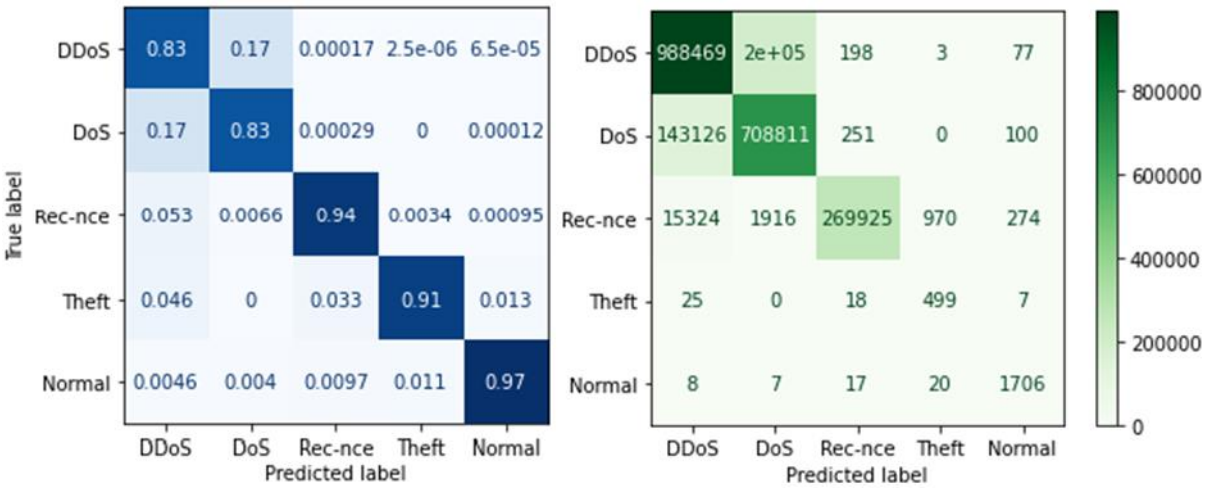


Figure 4. 15: Confusion matrices of averaging ensemble when $n = 3$.

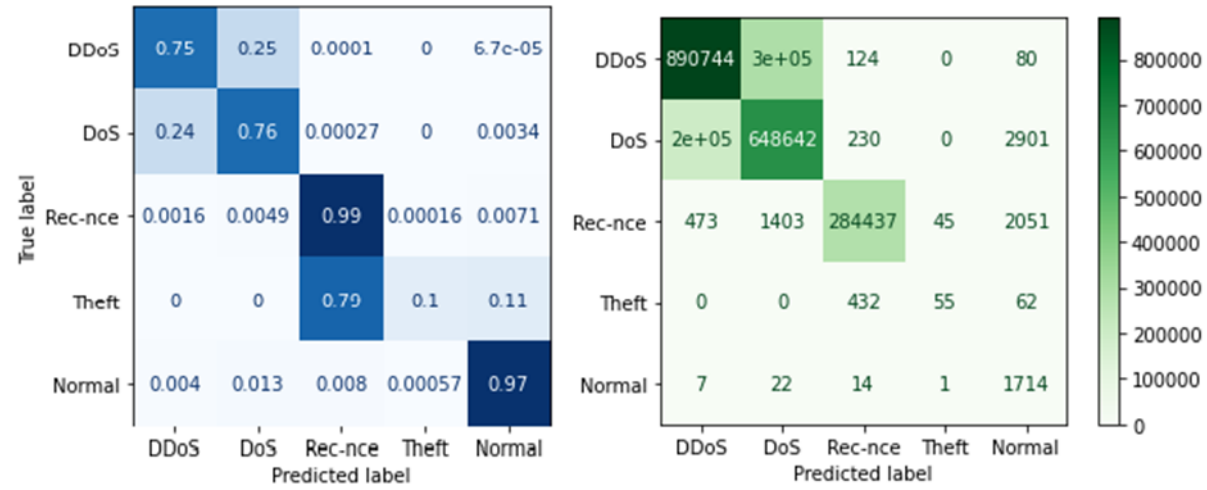


Figure 4. 16: Confusion matrices of averaging ensemble when $n = 6$.

4.6.2.2 Weighted Average

The models with excellent scores for classification in the weighted average ensemble contribute the largest proportion of the final predictions, and the other way around. In this case, to discover the best combination of weights that enhances the F1-score (the most significant metric in imbalanced data) a grid search method was applied. Table 4.12 presents a comparison of the results derived from the weighted average of $n = 3$ and $n = 6$ over every category found in the BoT-IoT dataset. In contrast to the averaging ensemble, the weighted average ensemble technique worked more effectively with $n = 6$, resulting

in the best F1-score in the tests we performed. Given that the F1 Score of Theft was 0.57%, the overall results for the Theft attack category were unsatisfactory despite the fact that the weighted average performed better than the averaging ensemble. This can be caused by the fact that the test data also have the class imbalance problem, making minority classes more vulnerable to false positive rates [162]. When False Negatives are more significant, sensitivity and G-Mean metrics are better suited to evaluate the weighted average ensemble approach. The Theft attack class performed highly in the situation of $n = 3$, scoring 0.91 and 0.95 for Sensitivity and G-Mean, respectively.

Figures 4.17 and 4.18 show the confusion matrices of a weighted average ensemble with $n = 3$ and $n = 6$. It is evident that, for $n = 6$, each of the five categories had excellent Sensitivity scores, with the exception of the DoS class categories, which did not do so well because 24% of DoS attacks were mistakenly identified as DDoS attacks. On the other hand, as false negatives and false positives are equally important to us in our work, we can say that the weighted average with $n = 6$ is the most effective intrusion detection technique.

Table 4. 12: The evaluation results of weighted average ensemble method

	Precision	Sensitivity	Specificity	F1 Score	G-Mean
Weights:	w1= 0.4, w2= 0.9, and w3= 0.8				
DDoS	0.86	0.99	0.83	0.92	0.91
DoS	0.98	0.78	0.99	0.87	0.88
Reconnaissance	0.99	0.99	1.00	0.99	0.99
Theft	0.34	0.91	1.00	0.49	0.95
Normal	0.79	0.97	1.00	0.87	0.99
macro avg	0.79	0.93	0.96	0.83	0.95
Weights :	w1= 0.1, w2= 0.1, w3= 0.6, w4= 0.1, w5= 0.1, and w6= 0.8				
DDoS	0.85	0.98	0.82	0.91	0.90
DoS	0.96	0.76	0.98	0.85	0.86
Reconnaissance	1.00	0.99	1.00	0.99	0.99
Theft	0.41	0.89	1.00	0.57	0.94
Normal	0.86	0.97	1.00	0.91	0.99
macro avg	0.82	0.92	0.96	0.85	0.94

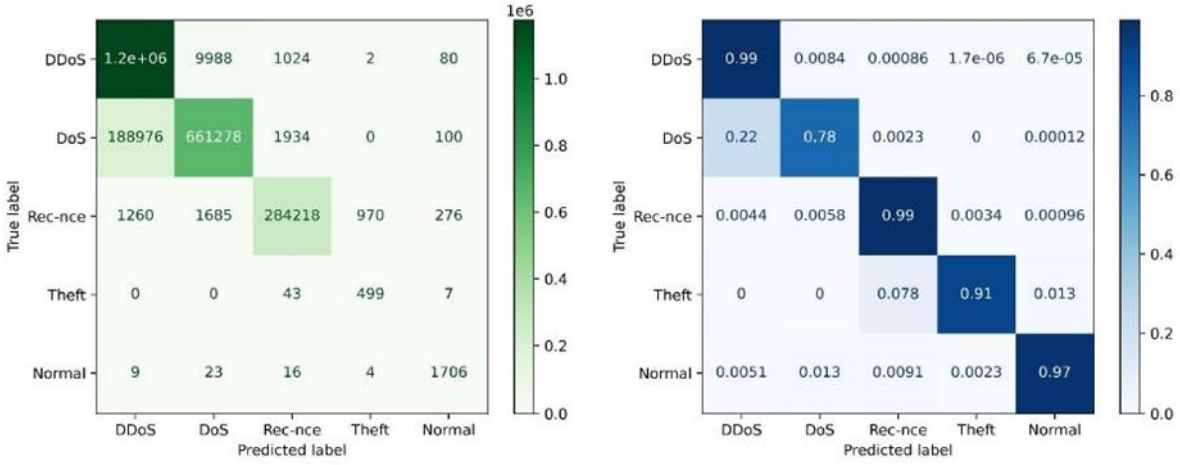


Figure 4. 17: Confusion matrices of weighted average ensemble when $n = 3$.

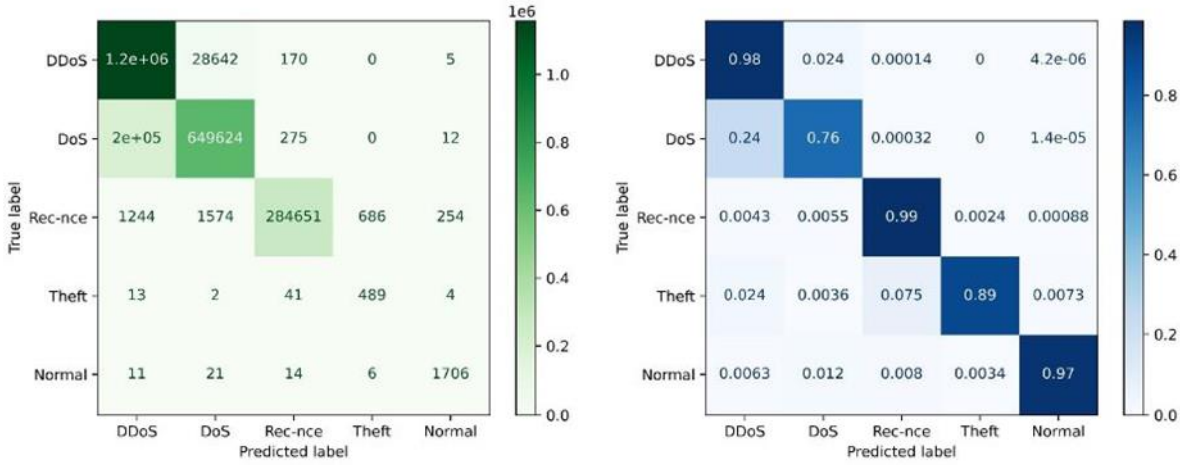


Figure 4. 18: Confusion matrices of weighted average ensemble when $n = 6$.

4.7 Discussion

The present section discusses our studies and their results for monitoring the condition of the road surface for three different types of roads: Real anomalies, man-made anomalies, and smooth roads. We also clarify the results of a big data system based on Apache Spark that was used to analyze network traffic.

Since every study used a private dataset with a variety of road types, it is impractical to compare performance with similar relevant studies in the area of RSC monitoring. However, we compare the best result from the three proposed frameworks to other

previous works that utilized DL models to demonstrate that our frameworks enhance the identification of anomalies on road surfaces.

Table 4.13 shows how our frameworks performed versus the studies [85, 91, 169] through the use of recall metric. The comparison shows that the three proposed frameworks outperform the other studies for recognizing smooth road surfaces, while Refs. [85] (Which identifies just potholes) and [169] achieved the best recall scores for recognizing real and man-made anomalies, respectively. As mentioned earlier, we contrast the results we obtained with the works that have been conducted using DL algorithms that had no training on smooth roads or real anomalies. Due to the fact that the authors of the present work used three types of road surfaces, they can compare reasonably only with Ref. [91]. As a result, we can say that it is evident that the three proposed frameworks detect every type of road surface condition better than Ref. [91].

Table 4. 13: Detection rate comparison with previous studies.

<table>
<tr><td>Reference</td><td>smooth road</td><td colspan="2">real anomalies</td><td colspan="2">man-made anomalies</td></tr>
<tr><td rowspan="2">Varona et al. [85]</td><td rowspan="2">-</td><td colspan="2">pothole</td><td>S. Hump</td><td>S. Gutter</td></tr>
<tr><td colspan="2">0.98</td><td>0.78</td><td>0.93</td></tr>
<tr><td rowspan="2">Setiawan et al. [91]</td><td rowspan="2">0.98</td><td>pothole</td><td>B. road</td><td colspan="2">S. Bump</td></tr>
<tr><td>0.86</td><td>0.62</td><td colspan="2">0.67</td></tr>
<tr><td rowspan="2">Kumar et al. [169]</td><td rowspan="2">0.96</td><td colspan="2" rowspan="2">-</td><td>R. Strip</td><td>S. Bump</td></tr>
<tr><td>0.97</td><td>0.98</td></tr>
<tr><td>First framework</td><td>0.99</td><td colspan="2">0.88</td><td colspan="2">0.88</td></tr>
<tr><td>Second framework</td><td>0.98</td><td colspan="2">0.85</td><td colspan="2">0.93</td></tr>
<tr><td>Third framework</td><td>1.00</td><td colspan="2">0.93</td><td colspan="2">0.95</td></tr>
</table>

Additionally, in order to detect intrusions in IoT networks, a new big data framework was proposed by this research. Table 4.14 compares our best results with those of earlier studies that used the entire BoT-IoT dataset in a big data environment. Although reference [151] included all features, even invalid ones, our method surpassed it in detecting Reconnaissance attacks, Theft, and Normal traffic. Since the authors in [152] skipped the assessment of theft attacks, did not disclose the accuracy of DDoS and DoS identification, and only used 10,000 DDoS and 10,000 DoS samples to evaluate normal traffic, we are unable to objectively compare our findings to the accuracy of reconnaissance attacks and normal traffic. In reference [153], the entire dataset was augmented, resulting in various augmented samples being included in both the training and testing sets. This led to

extremely high accuracy across all data classes, which is explained by the inflating of test accuracy. To avoid data leakage in the area of network traffic classification, test data must not be augmented. As a result, we assume that the findings found in reference [153] are not reliable. Through the use of the entire BoT-IoT dataset in the Apache Spark environment, this work aims to deliver trustworthy results for intrusion detection.

Table 4. 14: A comparison of the proposed approach with other related works using F1 Score metric.

Ref	DDoS	DoS	Reconnaissance	Theft	Normal
Manzano Sanchez et al.[153]	0.94	0.93	0.99	0.99	0.98
Abushwereb et al. [151]	0.999	0.991	0.888	0.232	0.718
Manzano S et al. [152]	-	-	0.999	-	0.983
Our approach	0.91	0.85	0.99	0.57	0.91

4.8 Conclusions

In this chapter, we presented our experimental evaluation and results for the proposed IoT frameworks, which we described in Chapters 2 and 3. The first proposed framework classified road anomalies into three types using hybrid deep learning models. The results of the experiments demonstrated that, when it came to predicting road anomalies the CNN-GRU model scored highest overall. By employing a combination of 3D hybrid deep learning models, the second proposed framework employed ensemble learning techniques to detect anomalies in road surfaces. The results of the experiments indicate that, when compared to the averaging ensemble and the other models, the weighted average ensemble provided the best overall performance.

The imbalance problem on the road data was resolved by the third proposed framework using multiple data augmentation methods. Comparing the DoppelGANger technique to SMOTE and Traditional techniques, it produced the best results with the 3D models. The class imbalance issue in the Bot-IoT dataset was solved in the fourth proposed framework through the combination of ensemble resampling and oversampling approaches. The final predictions were generated via the weighted average ensemble that showed good results and improved the results in minority classes.

The comparisons prove that the frameworks we have proposed in this thesis enhance the reliability of IoT data analysis while also increasing the accuracy of the results.

CONCLUSION AND FUTURE WORK

The Internet of Things (IoT) and artificial intelligence (AI) have opened up previously unthinkable possibilities by radically changing how we interact with the world and by utilizing the enormous potential of data. IoT applications are typically constructed using sensors that monitor the environment and then initiate actions to respond. Frequently, these actions involve changing external conditions. An IoT application with AI inference makes an effort to collect as much data as it can, simulating human senses. By combining AI with IoT, better and more effective systems are being created, revolutionizing our daily lives.

This thesis presents architectures for analyzing IoT data via AI algorithms. The main aim of the architectures that have been proposed is to analyze the vast amounts of valuable data that IoT devices generate. The thesis provides an architecture for an IoT intrusion detection system in an attempt to reduce the security threats that IoT devices face.

The architecture of the IoT system and its associated network were presented in Chapter 1. A summary of big data, IoT security, and IoT search engines was also provided. This chapter provided a quick introduction of the ML and DL algorithms used in IoT data classification.

Chapter 2 discussed three vibration-based RSC monitoring architectures that use hybrid deep learning algorithms for detecting anomalies in road surfaces. Moreover, the RSC-IoT and RSC datasets have been built and labeled in real time using Bluetooth and TCP/IP socket protocol. Then, we presented two architectures for RSC monitoring that take advantage of the IoT Search Engine (IoTSE) and cloud computing.

In Chapter 3, a novel Apache Spark-based big data architecture for detecting IoT network intrusions has been introduced. In order to do this, we first addressed the problem of class imbalance before using machine learning (ML) to classify data from the Bot-IoT dataset. Next, we utilize ensemble learning techniques to enhance the final predictions.

The proposed IoT architectures that were presented in Chapters 2 and 3 were examined in Chapter 4. The comparisons demonstrated that the hybrid deep learning and ensemble learning algorithms proposed in this thesis improved the reliability of IoT data analysis. Meanwhile, data augmentation techniques enhance the accuracy of results.

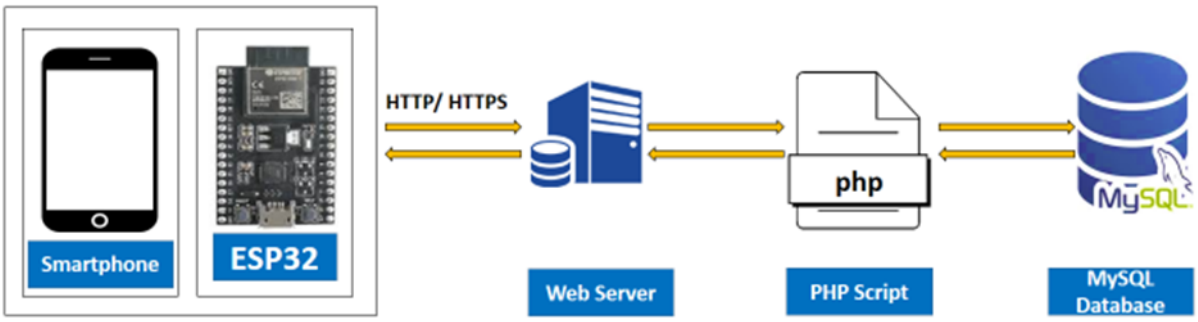
In this thesis, we have examined various IoT architectures, new and traditional approaches, and compared hybrid deep learning and ensemble learning methods with ML and DL techniques. The work in this thesis enabled us to present a multi-view assessment of IoT data analysis. In contrast to previous experimental methods, the results were satisfactory.

Although the applied algorithms employed in this thesis showed promising results, there is room for development in both IoT data collection and analysis in future work. In our future work, we intend to consider the following points:

- Evaluating the effectiveness of different hybrid deep learning models for identifying anomalies in the road surface.
- Employing more sophisticated ensemble learning techniques to enhance IoT data experiment results.
- Applying DL algorithms in a big data environment to identify intrusions on IoT networks.
- Implementing more advanced methods of data augmentation in order to achieve class balance in the IoT datasets.
- Examining the effect of vehicle velocity on road anomaly recognition using a dynamic sliding window technique.

APPENDIX A

ESP32/Smartphone Insert Data into MySQL Database using PHP



Using the approach depicted above, the ESP32/Smartphone establishes a connection with a web server using the HTTP protocol before moving on to a MySQL server. This approach steps are as follows:

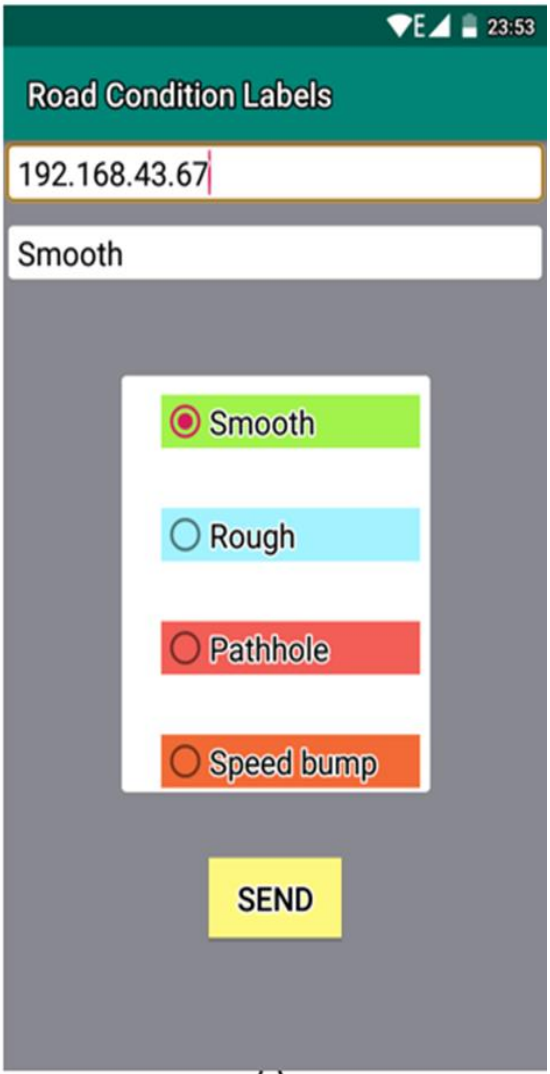
- The ESP32/Smartphone makes an HTTP request to the web server, including the data.
- A PHP script on the web server handles this request.
- Once the HTTP request has been processed, the PHP script gathers data and transfers it to MySQL.
- The PHP script uses an HTTP response to confirm whether or not the data has been received.

This is the interface of the Android smartphone application:

APPENDIX B

Real Time Data Labeling using TCP/IP socket/Bluetooth

The interfaces of the Android smartphone applications for data labeling utilizing (a) TCP/IP socket and (b) Bluetooth are as follows:



(a)

(b)

APPENDIX C

Extraction of CSV file for Road Surface Conditions from the RSC Dataset

```
id,time,xAccelValue,yAccelValue,zAccelValue,xGyroValue,yGyroValue
,zGyroValue,azimuth,pitch,roll,speed,latitude,longitude,altitude,
timer,freqency,paths,quality
1,2021-12-15 12:40:55.172,-
0.037110128,3.1196449,9.301474,0.010048735,-0.036975674,-
0.03613879,-0.93474793,-
0.3236003,0.003989683,0.0,36.5049543,2.87371073,229.035888671875,
0,50,via Parallele N1 G,Smooth
2,2021-12-15 12:40:55.250,-
0.4249708,2.9652188,9.076419,0.0014966214,0.004563161,-
0.017201971,-0.89847755,-
0.31544092,0.046787247,0.0,36.5049543,2.87371073,229.035888671875
,0,50,via Parallele N1 G,Smooth
3,2021-12-15 12:40:55.282,-
0.4249708,2.9652188,9.076419,0.0051618125,0.020445656,-
0.015369375,-0.89847755,-
0.31544092,0.046787247,0.0,36.5049543,2.87371073,229.035888671875
,0,50,via Parallele N1 G,Smooth
4,2021-12-15 12:40:55.328,-
0.39145198,2.5450363,9.074024,0.010048735,-0.0052106827,-
0.0007086089,-0.92730093,-
0.30215192,0.033760257,0.0,36.5049543,2.87371073,229.035888671875
,0,50,via Parallele N1 G,Smooth
5,2021-12-15 12:40:55.391,-0.46208093,2.8191724,9.169792,-
0.0058337618,-0.036975674,0.009065235,-0.9158473,-
0.29791358,0.050349057,0.0,36.5049543,2.87371073,229.035888671875
,0,50,via Parallele N1 G,Smooth
6,2021-12-15 12:40:55.422,-0.2657564,2.6396074,9.205706,-
0.010720683,-0.019260583,0.0060109096,-0.96277076,-
0.27913368,0.028860647,0.0,36.5049543,2.87371073,229.035888671875
,0,50,via Parallele N1 G,Smooth
```

APPENDIX D

Extraction of CSV file for Road Surface Conditions from the RSC-IoT Dataset

```
id,datetime,Ax,Ay,Az,Gx,Gy,Gz,quality
1,2023-08-05 11:31:39.179,-0.71,1.24,9.55,-0.05,0.02,0,smooth
2,2023-08-05 11:31:39.600,-0.58,1.35,9.39,-0.06,0.04,0,smooth
3,2023-08-05 11:31:39.897,-0.82,1.04,9.69,-0.07,0,0,smooth
4,2023-08-05 11:31:40.209,-0.76,0.95,10.7,-0.05,0.04,0.01,smooth
5,2023-08-05 11:31:40.505,-1.22,-0.04,9.62,-0.08,0.03,0,smooth
6,2023-08-05 11:31:40.817,-0.68,-0.18,10.41,-0.04,0,0.01,smooth
7,2023-08-05 11:31:41.145,-0.48,0.24,10.26,0,0.02,0.02,smooth
8,2023-08-05 11:31:41.441,-1.27,-0.04,10.33,-0.07,-0.02,0.01,smooth
9,2023-08-05 11:31:41.738,-0.79,0.47,9.87,-0.06,0.01,0.02,smooth
10,2023-08-05 11:31:42.065,-0.74,0.59,9.75,-0.01,-0.01,0.02,smooth
11,2023-08-05 11:31:42.362,-1.13,1.19,11.28,-0.02,0.03,0.01,smooth
12,2023-08-05 11:31:42.658,-1.19,0.81,9.49,-0.03,0,0,smooth
13,2023-08-05 11:31:42.970,-1.24,0.11,10.13,-0.07,0.02,0,smooth
14,2023-08-05 11:31:43.282,-1.98,-0.17,9.89,-0.04,0.04,0.05,smooth
15,2023-08-05 11:31:43.594,-2.75,-0.08,9.77,-0.07,-0.01,0.14,smooth
16,2023-08-05 11:31:43.906,-1.09,-0.64,10.52,-0.07,-0.02,0.14,smooth
17,2023-08-05 11:31:44.218,-1.58,-0.48,9.62,-0.04,-0.01,0.1,smooth
18,2023-08-05 11:31:44.530,-0.09,0.84,10.06,-0.06,-0.01,0.05,smooth
19,2023-08-05 11:31:44.811,-0.83,0.34,9.24,-0.05,0.02,0.02,smooth
20,2023-08-05 11:31:45.092,-0.3,0.68,10.73,-0.02,-0.01,-0.02,smooth
21,2023-08-05 11:31:45.372,-0.14,0.89,9.48,-0.05,0.03,-0.05,smooth
22,2023-08-05 11:31:45.638,-0.47,0.56,9.87,-0.04,0.04,-0.06,smooth
23,2023-08-05 11:31:45.918,-0.42,0.56,10.32,-0.06,0,-0.06,smooth
24,2023-08-05 11:31:46.184,-0.68,0.05,10.48,-0.03,0.02,-0.05,smooth
25,2023-08-05 11:31:46.464,-0.83,-0.18,10.78,-0.02,-0.03,-0.04,smooth
26,2023-08-05 11:31:46.730,-0.68,0.01,10.16,-0.02,-0,-0.03,smooth
27,2023-08-05 11:31:47.042,-1.13,-0.37,10.45,-0.05,-0.01,-0.02,smooth
28,2023-08-05 11:31:47.416,-1.19,-0.39,10.16,-0.06,0.01,0.01,smooth
29,2023-08-05 11:31:47.712,-0.97,-1.16,10.1,-0.05,-0.01,0.01,smooth
30,2023-08-05 11:31:48.071,-1.06,-1.4,10.29,-0.05,0.03,0.02,smooth
31,2023-08-05 11:31:48.508,-1.04,-1.3,9.94,-0.05,0,0.03,man-made
32,2023-08-05 11:31:48.820,-0.99,-0.14,9.72,-0.06,0.01,0.03,man-made
33,2023-08-05 11:31:49.132,-2.06,-1.49,11.66,0.08,0.1,0.03,man-made
34,2023-08-05 11:31:49.413,0.04,0.75,8.65,-0.07,-0.1,0.04,man-made
```

REFERENCES

[1] Y. Ismail, *Internet of Things (IoT) for Automated and Smart Applications*. BoD – Books on Demand, 2019.

[2] A. Kapoor, *Hands-On Artificial Intelligence for IoT: Expert machine learning and deep learning techniques for developing smarter IoT systems*. Packt Publishing Ltd, 2019.

[3] S. M. Ali *et al.*, « Drivers for Internet of Things (IoT) adoption in supply chains: Implications for sustainability in the post-pandemic era », *Computers & Industrial Engineering*, vol. 183, p. 109515, sept. 2023, doi: 10.1016/j.cie.2023.109515.

[4] R. Ande, B. Adebisi, M. Hammoudeh, et J. Saleem, « Internet of Things: Evolution and technologies from a security perspective », *Sustainable Cities and Society*, vol. 54, p. 101728, mars 2020, doi: 10.1016/j.scs.2019.101728.

[5] P. Sethi et S. R. Sarangi, « Internet of Things: Architectures, Protocols, and Applications », *Journal of Electrical and Computer Engineering*, vol. 2017, p. e9324035, janv. 2017, doi: 10.1155/2017/9324035.

[6] M. Mohammadi, A. Al-Fuqaha, S. Sorour, et M. Guizani, « Deep Learning for IoT Big Data and Streaming Analytics: A Survey », *IEEE Communications Surveys & Tutorials*, vol. 20, nº 4, p. 2923‑2960, 2018, doi: 10.1109/COMST.2018.2844341.

[7] R. A. Mouha, « Internet of Things (IoT) », *Journal of Data Analysis and Information Processing*, vol. 9, nº 2, Art. nº 2, mars 2021, doi: 10.4236/jdaip.2021.92006.

[8] W. Ejaz et A. Anpalagan, *Internet of Things for Smart Cities: Technologies, Big Data and Security*. Springer, 2021.

[9] S. M. Sidi Ahmed et S. Zulhuda, « The Concept of Internet of Things and its Challenges to Privacy ». Rochester, NY, 2015.

[10] C. Xenofontos, I. Zografopoulos, C. Konstantinou, A. Jolfaei, M. K. Khan, et K.-K. R. Choo, « Consumer, Commercial, and Industrial IoT (In)Security: Attack Taxonomy and Case Studies », *IEEE Internet of Things Journal*, vol. 9, nº 1, p. 199‑221, janv. 2022, doi: 10.1109/JIOT.2021.3079916.

[11] S. M. Tahsien, H. Karimipour, et P. Spachos, « Machine learning based solutions for security of Internet of Things (IoT): A survey », *Journal of Network and Computer Applications*, vol. 161, p. 102630, juill. 2020, doi: 10.1016/j.jnca.2020.102630.

[12] D. Jakhar et I. Kaur, « Artificial intelligence, machine learning and deep learning: definitions and differences », *Clinical and Experimental Dermatology*, vol. 45, nº 1, p. 131‑132, janv. 2020, doi: 10.1111/ced.14029.

[13] A. Dobson, K. Roy, X. Yuan, et J. Xu, « Performance Evaluation of Machine Learning Algorithms in Apache Spark for Intrusion Detection », in *2018 28th International Telecommunication Networks and Applications Conference (ITNAC)*, nov. 2018, p. 1‑6. doi: 10.1109/ATNAC.2018.8615314.

[14] K. K. Patel, S. M. Patel, et P. Scholar, « Internet of Things-IOT: Definition, Characteristics, Architecture, Enabling Technologies, Application & Future Challenges », 2016.

[15] Alexander S. Gillis, « What is IoT (Internet of Things) and How Does it Work? | Definition from TechTarget », IoT Agenda. Consulté le: 15 avril 2024. [Online]. Available on: https://www.techtarget.com/iotagenda/definition/Internet-of-Things-IoT

[16] Steve Ranger, « What is the IoT? Everything you need to know about the Internet of Things right now », ZDNET. Consulté le: 15 avril 2024. [Online]. Available on: https://www.zdnet.com/article/what-is-the-internet-of-things-everything-you-need-to-know-about-the-iot-right-now/

[17] « Understanding the Internet of Things (IoT) - GSMA / understanding-the-internet-of-things-iot-gsma.pdf / PDF4PRO », PDF4PRO. Consulté le: 15 avril 2024. [Online]. Available on: https://pdf4pro.com/view/understanding-the-internet-of-things-iot-gsma-735ca4.html

[18] A. Malik, T. Magar, H. Verma, M. Singh, et P. Sagar, « A Detailed Study Of An Internet Of Things (Iot) », *International Journal of Scientific & Technology Research*, vol. 8, p. 2989‑2994, janv. 2020.

[19] R. Heredia, « 4 Layers of IoT Architecture Explained ». Consulté le: 16 avril 2024. [Online]. Available on: https://www.zipitwireless.com/blog/4-layers-of-iot-architecture-explained

[20] A. Raj, « IoT Architecture - Detailed Explanation with Examples », AlmaBetter. Consulté le: 16 avril 2024. [Online]. Available on: https://www.almabetter.com/bytes/articles/iot-architecture

[21] M. Burhan, R. A. Rehman, B. Khan, et B.-S. Kim, « IoT Elements, Layered Architectures and Security Issues: A Comprehensive Survey », *Sensors*, vol. 18, nº 9, Art. nº 9, sept. 2018, doi: 10.3390/s18092796.

[22] L. José, « Unpacking IoT Architecture: Layers and Components Explained », Device Authority Ltd. Consulté le: 16 avril 2024. [Online]. Available on: https://www.deviceauthority.com/blog/unpacking-iot-architecture-layers-and-components-explained/

[23] Talal Sultan, « INTERNET OF THINGS-IOT: DEFINITION, ARCHITECTURE AND APPLICATIONS », *Egyptian Journal of Applied Science*, vol. 34, nº 1, p. 81‑95, févr. 2019, doi: 10.21608/ejas.2019.151723.

[24] A. Chandel, « IoT Gateway Architecture and Selection - SenseGrow ». Consulté le: 16 avril 2024. [Online]. Available on: https://www.sensegrow.com/blog/industrial-iot/iot-gateway-architecture-and-selection

[25] Z. Wu, K. Qiu, et J. Zhang, « A Smart Microcontroller Architecture for the Internet of Things », *Sensors*, vol. 20, nº 7, Art. nº 7, janv. 2020, doi: 10.3390/s20071821.

[26] J. Folkens, « Building a Gateway to the Internet of Things », *Texas*, 2014.

[27] A. Grygoruk et J. Legierski, « IoT gateway - implementation proposal based on Arduino board », in *2016 Federated Conference on Computer Science and Information Systems (FedCSIS)*, sept. 2016, p. 1011‑1014.

[28] J. Santos, J. J. P. C. Rodrigues, B. M. C. Silva, J. Casal, K. Saleem, et V. Denisov, « An IoT-based mobile gateway for intelligent personal assistants on mobile health environments », *Journal of Network and Computer Applications*, vol. 71, p. 194‑204, août 2016, doi: 10.1016/j.jnca.2016.03.014.

[29] N. Mohamudally, *Smartphones from an Applied Research Perspective*. BoD – Books on Demand, 2017.

[30] A. Gerodimos, L. Maglaras, M. A. Ferrag, N. Ayres, et I. Kantzavelou, « IoT: Communication protocols and security threats », *Internet of Things and Cyber-Physical Systems*, vol. 3, p. 1‑13, janv. 2023, doi: 10.1016/j.iotcps.2022.12.003.

[31] R. Vahidnia et F. J. Dian, *Cellular Internet of Things for Practitioners*. BRITISH COLUMBIA INSTITUTE OF TECHNOLOGY VANCOUVER, 2021.

[32] D. Law, D. Dove, J. D'Ambrosia, M. Hajduczenia, M. Laubach, et S. Carlson, « Evolution of ethernet standards in the IEEE 802.3 working group », *IEEE Communications Magazine*, vol. 51, nº 8, p. 88‑96, août 2013, doi: 10.1109/MCOM.2013.6576344.

[33] S. R. Borkar, « 7 - Long-term evolution for machines (LTE-M) », in *LPWAN Technologies for IoT and M2M Applications*, B. S. Chaudhari et M. Zennaro, Éd., Academic Press, 2020, p. 145‑166. doi: 10.1016/B978-0-12-818880-4.00007-7.

[34] A. Lavric et A. I. Petrariu, « LoRaWAN communication protocol: The new era of IoT », in *2018 International Conference on Development and Application Systems (DAS)*, mai 2018, p. 74‑77. doi: 10.1109/DAAS.2018.8396074.

[35] M. Gupta et S. Singh, « A Survey on the ZigBee Protocol, It's Security in Internet of Things (IoT) and Comparison of ZigBee with Bluetooth and Wi-Fi », in *Applications of Artificial Intelligence in Engineering*, X.-Z. Gao, R. Kumar, S. Srivastava, et B. P. Soni, Éd., Singapore: Springer, 2021, p. 473‑482. doi: 10.1007/978-981-33-4604-8_38.

[36] M. Sofi, « Bluetooth Protocol in Internet of Things (IoT), Security Challenges and a Comparison with Wi-Fi Protocol: A Review », *International Journal of Engineering Research and*, vol. V5, nov. 2016, doi: 10.17577/IJERTV5IS110266.

[37] S. Al-Sarawi, M. Anbar, K. Alieyan, et M. Alzubaidi, « Internet of Things (IoT) communication protocols: Review », in *2017 8th International Conference on Information Technology (ICIT)*, mai 2017, p. 685‑690. doi: 10.1109/ICITECH.2017.8079928.

[38] Ian Craggs, « MQTT Vs. HTTP for IoT ». Consulté le: 16 avril 2024. [Online]. Available on: https://www.hivemq.com/blog/mqtt-vs-http-protocols-in-iot-iiot/

[39] J. Norman et P. Joseph, « Security in Application Layer Protocols of IoT: Threats and Attacks », in *Security Breaches and Threat Prevention in the Internet of Things*, IGI Global, 2017, p. 76‑95. doi: 10.4018/978-1-5225-2296-6.ch004.

[40] A. Ludovici et A. Calveras, « A Proxy Design to Leverage the Interconnection of CoAP Wireless Sensor Networks with Web Applications », *Sensors*, vol. 15, nº 1, Art. nº 1, janv. 2015, doi: 10.3390/s150101217.

[41] G. M. B. Oliveira *et al.*, « Comparison Between MQTT and WebSocket Protocols for IoT Applications Using ESP8266 », in *2018 Workshop on Metrology for Industry 4.0 and IoT*, avr. 2018, p. 236‑241. doi: 10.1109/METROI4.2018.8428348.

[42] R. Krishnamurthi, A. Kumar, D. Gopinathan, A. Nayyar, et B. Qureshi, « An Overview of IoT Sensor Data Processing, Fusion, and Analysis Techniques », *Sensors*, vol. 20, nº 21, Art. nº 21, janv. 2020, doi: 10.3390/s20216076.

[43] S. Rautmare et D. M. Bhalerao, « MySQL and NoSQL database comparison for IoT application », in *2016 IEEE International Conference on Advances in Computer Applications (ICACA)*, oct. 2016, p. 235‑238. doi: 10.1109/ICACA.2016.7887957.

[44] G. Kiraz et C. Toğay, « IoT Data Storage: Relational & Non-Relational Database Management Systems Performance Comparison ».

[45] C. Asiminidis, G. Kokkonis, et S. Kontogiannis, « Database Systems Performance Evaluation for IoT Applications ». Rochester, NY, 1 décembre 2018. doi: 10.2139/ssrn.3360886.

[46] R. Alonso, R. Locci, et D. Reforgiato Recupero, « Improving digital twin experience through big data, IoT and social analysis: An architecture and a case study », *Heliyon*, vol. 10, nº 2, p. e24741, janv. 2024, doi: 10.1016/j.heliyon.2024.e24741.

[47] Y.-S. Kang, I.-H. Park, J. Rhee, et Y.-H. Lee, « MongoDB-Based Repository Design for IoT-Generated RFID/Sensor Big Data », *IEEE Sensors Journal*, vol. 16, nº 2, p. 485‑497, janv. 2016, doi: 10.1109/JSEN.2015.2483499.

[48] M. M. Eyada, W. Saber, M. M. El Genidy, et F. Amer, « Performance Evaluation of IoT Data Management Using MongoDB Versus MySQL Databases in Different Cloud Environments », *IEEE Access*, vol. 8, p. 110656‑110668, 2020, doi: 10.1109/ACCESS.2020.3002164.

[49] A. Duarte et J. Bernardino, « Cassandra for Internet of Things: An Experimental Evaluation », présenté à International Conference on Internet of Things and Big Data, SCITEPRESS, avr. 2016, p. 49‑56. doi: 10.5220/0005846400490056.

[50] S. Achari et R. Johari, « FOG-EE Computing: Fog, Edge and Elastic Computing, New Age Cloud Computing Paradigms », in *International Conference on Innovative Computing and Communications*, A. Khanna, D. Gupta, S. Bhattacharyya, A. E. Hassanien, S. Anand, et A. Jaiswal, Éd., Singapore: Springer, 2022, p. 579‑589. doi: 10.1007/978-981-16-3071-2_47.

[51] P. J. Escamilla-Ambrosio, A. Rodríguez-Mota, E. Aguirre-Anaya, R. Acosta-Bermejo, et M. Salinas-Rosales, « Distributing Computing in the Internet of Things: Cloud, Fog and Edge Computing Overview », in *NEO 2016: Results of the Numerical and Evolutionary Optimization Workshop NEO 2016 and the NEO Cities 2016 Workshop held on September 20-24, 2016 in Tlalnepantla, Mexico*, Y. Maldonado, L. Trujillo, O. Schütze, A. Riccardi, et M. Vasile, Éd., Cham: Springer International Publishing, 2018, p. 87‑115. doi: 10.1007/978-3-319-64063-1_4.

[52] M. Laroui, B. Nour, H. Moungla, M. A. Cherif, H. Afifi, et M. Guizani, « Edge and fog computing for IoT: A survey on current research activities & future directions », *Computer Communications*, vol. 180, p. 210‑231, déc. 2021, doi: 10.1016/j.comcom.2021.09.003.

[53] M. De Donno, K. Tange, et N. Dragoni, « Foundations and Evolution of Modern Computing Paradigms: Cloud, IoT, Edge, and Fog », *IEEE Access*, vol. 7, p. 150936‑150948, 2019, doi: 10.1109/ACCESS.2019.2947652.

[54] J. Wu, L. Ping, X. Ge, Y. Wang, et J. Fu, « Cloud Storage as the Infrastructure of Cloud Computing », in *2010 International Conference on Intelligent Computing and Cognitive Informatics*, juin 2010, p. 380‑383. doi: 10.1109/ICICCI.2010.119.

[55] K. Perumal et M. Manohar, « A Survey on Internet of Things: Case Studies, Applications, and Future Directions », in *Internet of Things: Novel Advances and Envisioned Applications*, D. P. Acharjya et M. K. Geetha, Éd., Cham: Springer International Publishing, 2017, p. 281‑297. doi: 10.1007/978-3-319-53472-5_14.

[56] N. Islam, M. M. Rashid, F. Pasandideh, B. Ray, S. Moore, et R. Kadel, « A Review of Applications and Communication Technologies for Internet of Things (IoT) and Unmanned Aerial Vehicle (UAV) Based Sustainable Smart Farming », *Sustainability*, vol. 13, n° 4, Art. n° 4, janv. 2021, doi: 10.3390/su13041821.

[57] C. Li, J. Wang, S. Wang, et Y. Zhang, « A review of IoT applications in healthcare », *Neurocomputing*, vol. 565, p. 127017, janv. 2024, doi: 10.1016/j.neucom.2023.127017.

[58] Vitaly Kurduban, « IoT in Manufacturing: Applications and Benefits of Smart Factories ». Consulté le: 17 avril 2024. [Online]. Available on: https://www.digi.com/blog/post/iot-in-manufacturing

[59] M. Soori, B. Arezoo, et R. Dastres, « Internet of things for smart factories in industry 4.0, a review », *Internet of Things and Cyber-Physical Systems*, vol. 3, p. 192‑204, janv. 2023, doi: 10.1016/j.iotcps.2023.04.006.

[60] M. Muntjir, M. Rahul, et H. A. Alhumyani, « An Analysis of Internet of Things (IoT): Novel Architectures, Modern Applications, Security Aspects and Future Scope with Latest Case Studies », *International Journal of Engineering Research*, vol. 6, nº 06, 2017.

[61] V. Rodriguez-Galiano, M. Sanchez-Castillo, M. Chica-Olmo, et M. Chica-Rivas, « Machine learning predictive models for mineral prospectivity: An evaluation of neural networks, random forest, regression trees and support vector machines », *Ore Geology Reviews*, vol. 71, p. 804‑818, déc. 2015, doi: 10.1016/j.oregeorev.2015.01.001.

[62] T. S. Madhulatha, « Comparison between K-Means and K-Medoids Clustering Algorithms », in *Advances in Computing and Information Technology*, D. C. Wyld, M. Wozniak, N. Chaki, N. Meghanathan, et D. Nagamalai, Éd., Berlin, Heidelberg: Springer, 2011, p. 472‑481. doi: 10.1007/978-3-642-22555-0_48.

[63] C. Joo, H. Kwon, J. Kim, H. Cho, et J. Lee, « Machine-learning-based optimization of operating conditions of naphtha cracking furnace to maximize plant profit », in *Computer Aided Chemical Engineering*, vol. 52, A. C. Kokossis, M. C. Georgiadis, et E. Pistikopoulos, Éd., in 33 European Symposium on Computer Aided Process Engineering, vol. 52. , Elsevier, 2023, p. 1397‑1402. doi: 10.1016/B978-0-443-15274-0.50222-5.

[64] G. Chassagnon, M. Vakalopolou, N. Paragios, et M.-P. Revel, « Deep learning: definition and perspectives for thoracic imaging », *Eur Radiol*, vol. 30, nº 4, p. 2021‑2030, avr. 2020, doi: 10.1007/s00330-019-06564-3.

[65] J. P. Bharadiya, « Exploring the Use of Recurrent Neural Networks for Time Series Forecasting », vol. 8, nº 5, 2023.

[66] R. Zhao, D. Wang, R. Yan, K. Mao, F. Shen, et J. Wang, « Machine Health Monitoring Using Local Feature-Based Gated Recurrent Unit Networks », *IEEE Transactions on Industrial Electronics*, vol. 65, nº 2, p. 1539‑1548, févr. 2018, doi: 10.1109/TIE.2017.2733438.

[67] I. K. Nti, A. F. Adekoya, et B. A. Weyori, « A comprehensive evaluation of ensemble learning for stock-market prediction », *J Big Data*, vol. 7, nº 1, p. 20, mars 2020, doi: 10.1186/s40537-020-00299-5.

[68] Y. Matsuzaka et Y. Uesawa, « Ensemble Learning, Deep Learning-Based and Molecular Descriptor-Based Quantitative Structure–Activity Relationships », *Molecules*, vol. 28, nº 5, Art. nº 5, janv. 2023, doi: 10.3390/molecules28052410.

[69] Z. A. Al-Sai *et al.*, « Explore Big Data Analytics Applications and Opportunities: A Review », *Big Data and Cognitive Computing*, vol. 6, nº 4, Art. nº 4, déc. 2022, doi: 10.3390/bdcc6040157.

[70] M. A. Amanullah *et al.*, « Deep learning and big data technologies for IoT security », *Computer Communications*, vol. 151, p. 495‑517, févr. 2020, doi: 10.1016/j.comcom.2020.01.016.

[71] W. Inoubli, S. Aridhi, H. Mezni, M. Maddouri, et E. Mephu Nguifo, « An experimental survey on big data frameworks », *Future Generation Computer Systems*, vol. 86, p. 546‑564, sept. 2018, doi: 10.1016/j.future.2018.04.032.

[72] E. Schiller, A. Aidoo, J. Fuhrer, J. Stahl, M. Ziörjen, et B. Stiller, « Landscape of IoT security », *Computer Science Review*, vol. 44, p. 100467, mai 2022, doi: 10.1016/j.cosrev.2022.100467.

[73] C. Qian, W. Gao, W. G. Hatcher, W. Liao, C. Lu, et W. Yu, « Search Engine for Heterogeneous Internet of Things Systems and Optimization », in *2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, août 2020, p. 475‑482. doi: 10.1109/DASC-PICom-CBDCom-CyberSciTech49142.2020.00089.

[74] F. Liang, C. Qian, W. G. Hatcher, et W. Yu, « Search Engine for the Internet of Things: Lessons From Web Search, Vision, and Opportunities », *IEEE Access*, vol. 7, p. 104673‑104691, 2019, doi: 10.1109/ACCESS.2019.2931659.

[75] H. Liang, L. Burgess, W. Liao, E. Blasch, et W. Yu, « Deep Learning Assist IoT Search Engine for Disaster Damage Assessment », *Cyber-Physical Systems*, vol. 9, nº 4, p. 313‑337, oct. 2023, doi: 10.1080/23335777.2022.2051210.

[76] W. G. Hatcher, C. Qian, W. Gao, F. Liang, K. Hua, et W. Yu, « Towards Efficient and Intelligent Internet of Things Search Engine », *IEEE Access*, vol. 9, p. 15778‑15795, 2021, doi: 10.1109/ACCESS.2021.3052759.

[77] H. V. Chand et J. Karthikeyan, « Survey on the Role of IoT in Intelligent Transportation System », vol. 11, nº 3, 2018.

[78] Jayna Locke, « IoT in Transportation: Solutions and Applications ». Consulté le: 18 avril 2024. [Online]. Available on: https://www.digi.com/blog/post/iot-solutions-for-transportation

[79] H. E. Hareru *et al.*, « The epidemiology of road traffic accidents and associated factors among drivers in Dilla Town, Southern Ethiopia », *Front Public Health*, vol. 10, p. 1007308, nov. 2022, doi: 10.3389/fpubh.2022.1007308.

[80] J. Ng, « Road accidents, is there a solution? », thesun.my. Consulté le: 18 avril 2024. [Online]. Available on: https://thesun.my/local_news/road-accidents-is-there-a-solution-HA10985460

[81] G. Baldini, R. Giuliani, et F. Geib, « On the Application of Time Frequency Convolutional Neural Networks to Road Anomalies' Identification with Accelerometers and Gyroscopes », *Sensors*, vol. 20, nº 22, Art. nº 22, janv. 2020, doi: 10.3390/s20226425.

[82] V. Toral, T. Krushangi, et V. H. R, « Automated potholes detection using vibration and vision-based techniques », *World Journal of Advanced Engineering Technology and Sciences*, vol. 10, nº 1, p. 157‑176, 2023, doi: 10.30574/wjaets.2023.10.1.0276.

[83] H.-J. Yang, H. Jang, et D.-S. Jeong, « Detection algorithm for road surface condition using wavelet packet transform and SVM », in *The 19th Korea-Japan Joint Workshop on Frontiers of Computer Vision*, janv. 2013, p. 323‑326. doi: 10.1109/FCV.2013.6485514.

[84] K. Higashimoto, H. Fukushima, K. Kamitani, et N. Chujo, « Identification of Road Surface Condition on Undeveloped Roads : — Aiming for Remote Car Driving — », in *2021 IEEE 10th Global Conference on Consumer Electronics (GCCE)*, oct. 2021, p. 777‑781. doi: 10.1109/GCCE53005.2021.9621967.

[85] B. Varona, A. Monteserin, et A. Teyseyre, « A deep learning approach to automatic road surface monitoring and pothole detection », *Pers Ubiquit Comput*, vol. 24, nº 4, p. 519‑534, août 2020, doi: 10.1007/s00779-019-01234-z.

[86] N. Silva, V. Shah, J. Soares, et H. Rodrigues, « Road Anomalies Detection System Evaluation », *Sensors*, vol. 18, nº 7, Art. nº 7, juill. 2018, doi: 10.3390/s18071984.

[87] J. Dib, K. Sirlantzis, et G. Howells, « A Review on Negative Road Anomaly Detection Methods », *IEEE Access*, vol. 8, p. 57298‑57316, 2020, doi: 10.1109/ACCESS.2020.2982220.

[88] S. Sattar, S. Li, et M. Chapman, « Developing a near real-time road surface anomaly detection approach for road surface monitoring », *Measurement*, vol. 185, p. 109990, nov. 2021, doi: 10.1016/j.measurement.2021.109990.

[89] A. Allouch, A. Koubâa, T. Abbes, et A. Ammar, « RoadSense: Smartphone Application to Estimate Road Conditions Using Accelerometer and Gyroscope », *IEEE*

Sensors Journal, vol. 17, nº 13, p. 4231‑4238, juill. 2017, doi: 10.1109/JSEN.2017.2702739.

[90] B. Zhou *et al.*, « Smartphone-based road manhole cover detection and classification », *Automation in Construction*, vol. 140, p. 104344, août 2022, doi: 10.1016/j.autcon.2022.104344.

[91] B. D. Setiawan, U. I. Serdült, et V. Kryssanov, « Smartphone Sensor Data Augmentation for Automatic Road Surface Assessment Using a Small Training Dataset », in *2021 IEEE International Conference on Big Data and Smart Computing (BigComp)*, janv. 2021, p. 239‑245. doi: 10.1109/BigComp51126.2021.00052.

[92] R. Du, G. Qiu, K. Gao, L. Hu, et L. Liu, « Abnormal Road Surface Recognition Based on Smartphone Acceleration Sensor », *Sensors*, vol. 20, nº 2, Art. nº 2, janv. 2020, doi: 10.3390/s20020451.

[93] C. Wu *et al.*, « An Automated Machine-Learning Approach for Road Pothole Detection Using Smartphone Sensor Data », *Sensors*, vol. 20, nº 19, Art. nº 19, janv. 2020, doi: 10.3390/s20195564.

[94] S. Sattar, S. Li, et M. Chapman, « Road Surface Monitoring Using Smartphone Sensors: A Review », *Sensors*, vol. 18, nº 11, Art. nº 11, nov. 2018, doi: 10.3390/s18113845.

[95] Ankita, S. Rani, H. Babbar, S. Coleman, A. Singh, et H. M. Aljahdali, « An Efficient and Lightweight Deep Learning Model for Human Activity Recognition Using Smartphones », *Sensors*, vol. 21, nº 11, Art. nº 11, janv. 2021, doi: 10.3390/s21113845.

[96] M. Jalayer, C. Orsenigo, et C. Vercellis, « Fault detection and diagnosis for rotating machinery: A model based on convolutional LSTM, Fast Fourier and continuous wavelet transforms », *Computers in Industry*, vol. 125, p. 103378, févr. 2021, doi: 10.1016/j.compind.2020.103378.

[97] M. Farooq et E. Sazonov, « Detection of chewing from piezoelectric film sensor signals using ensemble classifiers », in *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, août 2016, p. 4929‑4932. doi: 10.1109/EMBC.2016.7591833.

[98] A. Prasad, A. K. Tyagi, M. M. Althobaiti, A. Almulihi, R. F. Mansour, et A. M. Mahmoud, « Human Activity Recognition Using Cell Phone-Based Accelerometer and Convolutional Neural Network », *Applied Sciences*, vol. 11, nº 24, Art. nº 24, janv. 2021, doi: 10.3390/app112412099.

[99] N. Dua, S. N. Singh, et V. B. Semwal, « Multi-input CNN-GRU based human activity recognition using wearable sensors », *Computing*, vol. 103, nº 7, p. 1461‑1478, juill. 2021, doi: 10.1007/s00607-021-00928-8.

[100] H. Kutlu et E. Avcı, « A Novel Method for Classifying Liver and Brain Tumors Using Convolutional Neural Networks, Discrete Wavelet Transform and Long Short-Term Memory Networks », *Sensors*, vol. 19, nº 9, Art. nº 9, janv. 2019, doi: 10.3390/s19091992.

[101] K. Lee, J.-K. Kim, J. Kim, K. Hur, et H. Kim, « CNN and GRU combination scheme for Bearing Anomaly Detection in Rotating Machinery Health Monitoring », in *2018 1st IEEE International Conference on Knowledge Innovation and Invention (ICKII)*, juill. 2018, p. 102‑105. doi: 10.1109/ICKII.2018.8569155.

[102] M. Sajjad *et al.*, « A Novel CNN-GRU-Based Hybrid Approach for Short-Term Residential Load Forecasting », *IEEE Access*, vol. 8, p. 143759‑143768, 2020, doi: 10.1109/ACCESS.2020.3009537.

[103] Md. Z. Islam, Md. M. Islam, et A. Asraf, « A combined deep CNN-LSTM network for the detection of novel coronavirus (COVID-19) using X-ray images », *Informatics in Medicine Unlocked*, vol. 20, p. 100412, janv. 2020, doi: 10.1016/j.imu.2020.100412.

[104] X. Chen, X. Xie, et D. Teng, « Short-term Traffic Flow Prediction Based on ConvLSTM Model », in *2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC)*, juin 2020, p. 846‑850. doi: 10.1109/ITOEC49072.2020.9141783.

[105] A. Zekry, A. Sayed, M. Moussa, et M. Elhabiby, « Anomaly Detection using IoT Sensor-Assisted ConvLSTM Models for Connected Vehicles », in *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, avr. 2021, p. 1‑6. doi: 10.1109/VTC2021-Spring51267.2021.9449086.

[106] S. Garg et N. Baliyan, « Comparative analysis of Android and iOS from security viewpoint », *Computer Science Review*, vol. 40, p. 100372, mai 2021, doi: 10.1016/j.cosrev.2021.100372.

[107] W. Gomolka, « The concept of Sockets and basic Function Blocks for communication over Ethernet Part 2 TCP Server and TCP Client », juill. 2014.

[108] M. Malekzadeh, R. G. Clegg, A. Cavallaro, et H. Haddadi, « Mobile sensor data anonymization », in *Proceedings of the International Conference on Internet of Things Design and Implementation*, in IoTDI '19. New York, NY, USA: Association for Computing Machinery, avr. 2019, p. 49‑58. doi: 10.1145/3302505.3310068.

[109] M. Chauhan, P. Thorwe, M. J. Mukherjee, et Y. S. Rao, « Sensor Data Analysis Using Moving Average Filter and 256-Point FFT for Wireless Sensor Networks », in *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, juill. 2018, p. 1‑7. doi: 10.1109/ICCCNT.2018.8493974.

[110] W. Liu, S. Liu, Y. Wang, G. Li, et L. Yu, « Research on Grain Pile Temperature Prediction Based on CNN-GRU Neural Network », in *Advances in Intelligent Systems, Computer Science and Digital Economics III*, Z. Hu, S. Gavriushin, S. Petoukhov, et

M. He, Éd., Cham: Springer International Publishing, 2022, p. 214‑226. doi: 10.1007/978-3-030-97057-4_19.

[111] J. Wang, J. Jiao, L. Bao, S. He, Y. Liu, et W. Liu, « Self-Supervised Spatio-Temporal Representation Learning for Videos by Predicting Motion and Appearance Statistics », présenté à Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, p. 4006‑4015.

[112] W. Saadeh, S. A. Butt, et M. A. B. Altaf, « A Patient-Specific Single Sensor IoT-Based Wearable Fall Prediction and Detection System », *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 27, nº 5, p. 995‑1003, mai 2019, doi: 10.1109/TNSRE.2019.2911602.

[113] H. Qiao, T. Wang, P. Wang, S. Qiao, et L. Zhang, « A Time-Distributed Spatiotemporal Feature Learning Method for Machine Health Monitoring with Multi-Sensor Time Series », *Sensors*, vol. 18, nº 9, Art. nº 9, sept. 2018, doi: 10.3390/s18092932.

[114] S. Montaha, S. Azam, A. K. M. R. H. Rafid, Md. Z. Hasan, A. Karim, et A. Islam, « TimeDistributed-CNN-LSTM: A Hybrid Approach Combining CNN and LSTM to Classify Brain Tumor on 3D MRI Scans Performing Ablation Study », *IEEE Access*, vol. 10, p. 60039‑60059, 2022, doi: 10.1109/ACCESS.2022.3179577.

[115] J. Wang, T. Zhu, J. Gan, L. L. Chen, H. Ning, et Y. Wan, « Sensor Data Augmentation by Resampling in Contrastive Learning for Human Activity Recognition », *IEEE Sensors Journal*, vol. 22, nº 23, p. 22994‑23008, déc. 2022, doi: 10.1109/JSEN.2022.3214198.

[116] C. Oh, S. Han, et J. Jeong, « Time-Series Data Augmentation based on Interpolation », *Procedia Computer Science*, vol. 175, p. 64‑71, janv. 2020, doi: 10.1016/j.procs.2020.07.012.

[117] K. M. Rashid et J. Louis, « Times-series data augmentation and deep learning for construction equipment activity recognition », *Advanced Engineering Informatics*, vol. 42, p. 100944, oct. 2019, doi: 10.1016/j.aei.2019.100944.

[118] J. A. Sáez, J. Luengo, J. Stefanowski, et F. Herrera, « SMOTE–IPF: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering », *Information Sciences*, vol. 291, p. 184‑203, janv. 2015, doi: 10.1016/j.ins.2014.08.051.

[119] A. Fernandez, S. Garcia, F. Herrera, et N. V. Chawla, « SMOTE for Learning from Imbalanced Data: Progress and Challenges, Marking the 15-year Anniversary », *Journal of Artificial Intelligence Research*, vol. 61, p. 863‑905, avr. 2018, doi: 10.1613/jair.1.11192.

[120] Y. Yang, X. Yang, W. Tang, et L. Li, « A Undersampling-DoppelGANger based Data Generation Method for Unbalanced BGP Data », in *2023 IEEE 9th International*

Conference on Cloud Computing and Intelligent Systems (CCIS), août 2023, p. 100‑105. doi: 10.1109/CCIS59572.2023.10263221.

[121] X. Cai *et al.*, « Data Self-Expansion and DoppelGANger-Based Time-Series Modeling for Realistic Steam Data Generation », in *2023 8th International Conference on Power and Renewable Energy (ICPRE)*, sept. 2023, p. 1969‑1974. doi: 10.1109/ICPRE59655.2023.10353886.

[122] A. Khraisat et A. Alazab, « A critical review of intrusion detection systems in the internet of things: techniques, deployment strategy, validation strategy, attacks, public datasets and challenges », *Cybersecur*, vol. 4, n° 1, p. 18, mars 2021, doi: 10.1186/s42400-021-00077-7.

[123] J. Lansky *et al.*, « Deep Learning-Based Intrusion Detection Systems: A Systematic Review », *IEEE Access*, vol. 9, p. 101574‑101599, 2021, doi: 10.1109/ACCESS.2021.3097247.

[124] T. Cruz et P. Simoes, *ECCWS 2019 18th European Conference on Cyber Warfare and Security*. Academic Conferences and publishing limited, 2019.

[125] G. De La Torre Parra, P. Rad, K.-K. R. Choo, et N. Beebe, « Detecting Internet of Things attacks using distributed deep learning », *Journal of Network and Computer Applications*, vol. 163, p. 102662, août 2020, doi: 10.1016/j.jnca.2020.102662.

[126] L. Santos, C. Rabadão, et R. Gonçalves, « Flow Monitoring System for IoT Networks », in *New Knowledge in Information Systems and Technologies*, Á. Rocha, H. Adeli, L. P. Reis, et S. Costanzo, Éd., Cham: Springer International Publishing, 2019, p. 420‑430. doi: 10.1007/978-3-030-16184-2_40.

[127] S. Feng, J. Keung, X. Yu, Y. Xiao, et M. Zhang, « Investigation on the stability of SMOTE-based oversampling techniques in software defect prediction », *Information and Software Technology*, vol. 139, p. 106662, nov. 2021, doi: 10.1016/j.infsof.2021.106662.

[128] I. Jung, J. Ji, et C. Cho, « EmSM: Ensemble Mixed Sampling Method for Classifying Imbalanced Intrusion Detection Data », *Electronics*, vol. 11, n° 9, Art. n° 9, janv. 2022, doi: 10.3390/electronics11091346.

[129] P. Gogoi, M. H. Bhuyan, D. K. Bhattacharyya, et J. K. Kalita, « Packet and Flow Based Network Intrusion Dataset », in *Contemporary Computing*, M. Parashar, D. Kaushik, O. F. Rana, R. Samtaney, Y. Yang, et A. Zomaya, Éd., Berlin, Heidelberg: Springer, 2012, p. 322‑334. doi: 10.1007/978-3-642-32129-0_34.

[130] T. Saranya, S. Sridevi, C. Deisy, T. D. Chung, et M. K. A. A. Khan, « Performance Analysis of Machine Learning Algorithms in Intrusion Detection System: A Review », *Procedia Computer Science*, vol. 171, p. 1251‑1260, janv. 2020, doi: 10.1016/j.procs.2020.04.133.

[131] H.-J. Liao, C.-H. Richard Lin, Y.-C. Lin, et K.-Y. Tung, « Intrusion detection system: A comprehensive review », *Journal of Network and Computer Applications*, vol. 36, nº 1, p. 16‑24, janv. 2013, doi: 10.1016/j.jnca.2012.09.004.

[132] S. García, S. Ramírez-Gallego, J. Luengo, J. M. Benítez, et F. Herrera, « Big data preprocessing: methods and prospects », *Big Data Anal*, vol. 1, nº 1, p. 9, nov. 2016, doi: 10.1186/s41044-016-0014-0.

[133] S. A. Abdulkareem, C. H. Foh, H. Lee, F. Carrez, et K. Moessner, « IoT Network Intrusion Detection with Ensemble Learners », in *2022 13th International Conference on Information and Communication Technology Convergence (ICTC)*, oct. 2022, p. 510‑514. doi: 10.1109/ICTC55196.2022.9952376.

[134] J. L. Leevy, J. Hancock, T. M. Khoshgoftaar, et J. M. Peterson, « IoT information theft prediction using ensemble feature selection », *J Big Data*, vol. 9, nº 1, p. 6, janv. 2022, doi: 10.1186/s40537-021-00558-z.

[135] I. Vaccari, G. Chiola, M. Aiello, M. Mongelli, et E. Cambiaso, « MQTTset, a New Dataset for Machine Learning Techniques on MQTT », *Sensors*, vol. 20, nº 22, Art. nº 22, janv. 2020, doi: 10.3390/s20226578.

[136] H. Hindy, E. Bayne, M. Bures, R. Atkinson, C. Tachtatzis, et X. Bellekens, « Machine Learning Based IoT Intrusion Detection System: An MQTT Case Study (MQTT-IoT-IDS2020 Dataset) », in *Selected Papers from the 12th International Networking Conference*, B. Ghita et S. Shiaeles, Éd., Cham: Springer International Publishing, 2021, p. 73‑84. doi: 10.1007/978-3-030-64758-2_6.

[137] H. Kang, Ahn, Dong Hyun, Lee, Gyung Min, Yoo, Jeong Do, Park, Kyung Ho, et Kim, Huy Kang, « IoT network intrusion dataset ». IEEE, 27 septembre 2019. doi: 10.21227/q70p-q449.

[138] S. Garcia, A. Parmisano, et M. J. Erquiaga, « IoT-23: A labeled dataset with malicious and benign IoT network traffic ». Zenodo, 20 janvier 2020. doi: 10.5281/zenodo.4743746.

[139] A. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood, et A. Anwar, « TON_IoT Telemetry Dataset: A New Generation Dataset of IoT and IIoT for Data-Driven Intrusion Detection Systems », *IEEE Access*, vol. 8, p. 165130‑165150, 2020, doi: 10.1109/ACCESS.2020.3022862.

[140] N. Moustafa et J. Slay, « UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set) », in *2015 Military Communications and Information Systems Conference (MilCIS)*, nov. 2015, p. 1‑6. doi: 10.1109/MilCIS.2015.7348942.

[141] S. Choudhary et N. Kesswani, « A Survey: Intrusion Detection Techniques for Internet of Things », *IJISP*, vol. 13, nº 1, p. 86‑105, janv. 2019, doi: 10.4018/IJISP.2019010107.

[142] Y. Kayode Saheed, A. Idris Abiodun, S. Misra, M. Kristiansen Holone, et R. Colomo-Palacios, « A machine learning-based intrusion detection for detecting internet of things network attacks », *Alexandria Engineering Journal*, vol. 61, nº 12, p. 9395‑9409, déc. 2022, doi: 10.1016/j.aej.2022.02.063.

[143] T. Saba, A. Rehman, T. Sadad, H. Kolivand, et S. A. Bahaj, « Anomaly-based intrusion detection system for IoT networks through deep learning model », *Computers and Electrical Engineering*, vol. 99, p. 107810, avr. 2022, doi: 10.1016/j.compeleceng.2022.107810.

[144] K. Albulayhi, Q. Abu Al-Haija, S. A. Alsuhibany, A. A. Jillepalli, M. Ashrafuzzaman, et F. T. Sheldon, « IoT Intrusion Detection Using Machine Learning with a Novel High Performing Feature Selection Method », *Applied Sciences*, vol. 12, nº 10, Art. nº 10, janv. 2022, doi: 10.3390/app12105015.

[145] J. Simon, N. Kapileswar, P. K. Polasi, et M. A. Elaveini, « Hybrid intrusion detection system for wireless IoT networks using deep learning algorithm », *Computers and Electrical Engineering*, vol. 102, p. 108190, sept. 2022, doi: 10.1016/j.compeleceng.2022.108190.

[146] A. K. Sahu, S. Sharma, M. Tanveer, et R. Raja, « Internet of Things attack detection using hybrid Deep Learning Model », *Computer Communications*, vol. 176, p. 146‑154, août 2021, doi: 10.1016/j.comcom.2021.05.024.

[147] Y. Zhang et Q. Liu, « On IoT intrusion detection based on data augmentation for enhancing learning on unbalanced samples », *Future Generation Computer Systems*, vol. 133, p. 213‑227, août 2022, doi: 10.1016/j.future.2022.03.007.

[148] M. A. Ferrag, O. Friha, L. Maglaras, H. Janicke, et L. Shu, « Federated Deep Learning for Cyber Security in the Internet of Things: Concepts, Applications, and Experimental Analysis », *IEEE Access*, vol. 9, p. 138509‑138542, 2021, doi: 10.1109/ACCESS.2021.3118642.

[149] A. Demirpolat, A. K. Sarica, et P. Angin, « ProtÉdge: A few-shot ensemble learning approach to software-defined networking-assisted edge security », *Transactions on Emerging Telecommunications Technologies*, vol. 32, nº 6, p. e4138, 2021, doi: 10.1002/ett.4138.

[150] R. A. Ramadan et K. Yadav, « A Novel Hybrid Intrusion Detection System (IDS) for the Detection of Internet of Things (IoT) Network Attacks ». Rochester, NY, 20 décembre 2020.

[151] M. Abushwereb, M. Alkasassbeh, M. Almseidin, et M. Mustafa, « An accurate IoT Intrusion Detection Framework using Apache Spark ». arXiv, 21 février 2022. doi: 10.48550/arXiv.2203.04347.

[152] R. Manzano S., N. Goel, M. Zaman, R. Joshi, et K. Naik, « Design of a Machine Learning Based Intrusion Detection Framework and Methodology for IoT Networks »,

in *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*, janv. 2022, p. 0191‑0198. doi: 10.1109/CCWC54503.2022.9720857.

[153] R. A. Manzano Sanchez, M. Zaman, N. Goel, K. Naik, et R. Joshi, « Towards Developing a Robust Intrusion Detection Model Using Hadoop–Spark and Data Augmentation for IoT Networks », *Sensors*, vol. 22, nº 20, Art. nº 20, janv. 2022, doi: 10.3390/s22207726.

[154] N. Koroniotis, N. Moustafa, E. Sitnikova, et B. Turnbull, « Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset », *Future Generation Computer Systems*, vol. 100, p. 779‑796, nov. 2019, doi: 10.1016/j.future.2019.05.041.

[155] J. M. Peterson, J. L. Leevy, et T. M. Khoshgoftaar, « A Review and Analysis of the Bot-IoT Dataset », in *2021 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, août 2021, p. 20‑27. doi: 10.1109/SOSE52839.2021.00007.

[156] S. Salloum, R. Dautov, X. Chen, P. X. Peng, et J. Z. Huang, « Big data analytics on Apache Spark », *Int J Data Sci Anal*, vol. 1, nº 3, p. 145‑164, nov. 2016, doi: 10.1007/s41060-016-0027-9.

[157] K. Wang et M. M. H. Khan, « Performance Prediction for Apache Spark Platform », in *2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems*, août 2015, p. 166‑173. doi: 10.1109/HPCC-CSS-ICESS.2015.246.

[158] J. P.S. et P. Samuel, « Analysis and Modeling of Resource Management Overhead in Hadoop YARN Clusters », in *2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech)*, nov. 2017, p. 957‑964. doi: 10.1109/DASC-PICom-DataCom-CyberSciTec.2017.159.

[159] R. DelValle, P. Kaushik, A. Jain, J. Hartog, et M. Govindaraju, « Electron: Towards Efficient Resource Management on Heterogeneous Clusters with Apache Mesos », in *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, juin 2017, p. 262‑269. doi: 10.1109/CLOUD.2017.41.

[160] X. Meng *et al.*, « MLlib: Machine Learning in Apache Spark », *Journal of Machine Learning Research*, vol. 17, nº 34, p. 1‑7, 2016.

[161] E. Bisong, « Google Colaboratory », in *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*, E. Bisong, Éd., Berkeley, CA: Apress, 2019, p. 59‑64. doi: 10.1007/978-1-4842-4470-8_7.

[162] V. H. Alves Ribeiro et G. Reynoso-Meza, « Ensemble learning by means of a multi-objective optimization design approach for dealing with imbalanced data sets », *Expert Systems with Applications*, vol. 147, p. 113232, juin 2020, doi: 10.1016/j.eswa.2020.113232.

[163] S. Bagui et K. Li, « Resampling imbalanced data for network intrusion detection datasets », *J Big Data*, vol. 8, nº 1, p. 6, janv. 2021, doi: 10.1186/s40537-020-00390-x.

[164] arvindpdmn Pawan_Dubey, « Confusion Matrix », Devopedia. Consulté le: 19 avril 2024. [Online]. Available on: https://devopedia.org/confusion-matrix

[165] A. Hadj-Attou, Y. Kabir, et F. Ykhlef, « Hybrid deep learning models for road surface condition monitoring », *Measurement*, vol. 220, p. 113267, oct. 2023, doi: 10.1016/j.measurement.2023.113267.

[166] S. Y. Yerima, M. K. Alzaylaee, A. Shajan, et V. P, « Deep Learning Techniques for Android Botnet Detection », *Electronics*, vol. 10, nº 4, Art. nº 4, janv. 2021, doi: 10.3390/electronics10040519.

[167] T. Wang, J. Li, M. Zhang, A. Zhu, H. Snoussi, et C. Choi, « An enhanced 3DCNN-ConvLSTM for spatiotemporal multimedia data analysis », *Concurrency and Computation: Practice and Experience*, vol. 33, nº 2, p. e5302, 2021, doi: 10.1002/cpe.5302.

[168] H.-X. Gao, S. Kuenzel, et X.-Y. Zhang, « A Hybrid ConvLSTM-Based Anomaly Detection Approach for Combating Energy Theft », *IEEE Transactions on Instrumentation and Measurement*, vol. 71, p. 1‑10, 2022, doi: 10.1109/TIM.2022.3201569.

[169] T. Kumar, D. Acharya, et D. Lohani, « Modeling IoT Enabled Classification System for Road Surface Monitoring », in *2022 14th International Conference on COMmunication Systems & NETworkS (COMSNETS)*, janv. 2022, p. 836‑841. doi: 10.1109/COMSNETS53615.2022.9668507.