Burak KOLUKISA

A Ph.D. Thesis

AGU 2024

# MACHINE LEARNING APPROACHES FOR INTERNET OF THINGS BASED VEHICLE TYPE CLASSIFICATION AND NETWORK ANOMALY DETECTION

A THESIS
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
AND THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE OF ABDULLAH GUL UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

By
Burak KOLUKISA
January 2024

# MACHINE LEARNING APPROACHES FOR INTERNET OF THINGS BASED VEHICLE TYPE CLASSIFICATION AND NETWORK ANOMALY DETECTION

A THESIS
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
AND THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE OF ABDULLAH GUL UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

By
Burak KOLUKISA

January 2024

## SCIENTIFIC ETHICS COMPLIANCE

I hereby declare that all information in this document has been obtained in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all materials and results that are not original to this work.

Name-Surname: Burak KOLUKISA

Signature:

## REGULATORY COMPLIANCE

Ph.D. thesis titled “Machine Learning Approaches for Internet of Things Based Vehicle Type Classification and Network Anomaly Detection” has been prepared in accordance with the Thesis Writing Guidelines of the Abdullah Gül University, Graduate School of Engineering and Science.

| Prepared By | Advisor |
| --- | --- |
| Burak KOLUKISA | Prof. V. Çağrı GÜNGÖR |

Head of the Electrical and Computer Engineering Program

Asst. Prof. Samet GÜLER

## ACCEPTANCE AND APPROVAL

Ph.D. thesis titled “Machine Learning Approaches for Internet of Things Based Vehicle Type Classification and Network Anomaly Detection” prepared by Burak KOLUKISA has been accepted by the jury in the Electrical and Computer Engineering Graduate Program at Abdullah Gül University, Graduate School of Engineering and Science.

…../ …../ 2024

**JURY:**

Advisor: Prof. V. Çağrı GÜNGÖR

Member: Prof. Celal ÖZTÜRK

Member: Assoc. Prof. Özkan Ufuk NALBANTOĞLU

Member: Asst. Prof. Abdulkadir KÖSE

Member: Asst. Prof. Rıfat KURBAN

**APPROVAL:**

The acceptance of this Ph.D. thesis has been approved by the decision of the Abdullah Gül University, Graduate School of Engineering and Science, Executive Board dated …../…../ …….. and numbered ……..……. .

…….. /…….. / ……..

**(Date)**

Graduate School Dean
Prof. İrfan ALAN

# ABSTRACT

## MACHINE LEARNING APPROACHES FOR INTERNET OF THINGS BASED VEHICLE TYPE CLASSIFICATION AND NETWORK ANOMALY DETECTION

Burak KOLUKISA
Ph.D. in Electrical and Computer Engineering
Advisor: Prof. V. Çağrı GÜNGÖR
January 2024

This thesis presents innovative approaches in the realms of Intelligent Transportation Systems (ITS) and Network Intrusion Detection Systems (NIDS) within the Internet of Things (IoT). Leveraging IoT technologies, a low-cost, battery-operated 3-D magnetic sensor has been developed for ITS to enable the classification of vehicle categories. The research presents machine learning and deep learning models that are improved by using oversampling, feature selection and extraction methods, hyperparameter optimization, and converting signals into 2-D images. New methods have been proposed for vehicle type classification to boost classification performance and achieve an accuracy of up to 92.92%. Additionally, the increasing reliance on IoT devices for such applications introduces significant cybersecurity risks. To mitigate these vulnerabilities, a novel logistic regression model trained with a parallel artificial bee colony (LR-ABC) algorithm has been proposed for network anomaly detection. This model incorporates hyperparameter optimization to enhance detection capabilities, showcasing superior performance on popular benchmark NIDS datasets with accuracies of 88.25% and 90.11%. Overall, this research contributes to the advancement of IoT and IoT cybersecurity by offering robust, scalable, and efficient solutions. These innovations not only enhance vehicle type classification and network security in the IoT era but also pave the way for future IoT infrastructure development in an increasingly connected digital landscape.

*Keywords: Internet of Things (IoT), Intelligent Transportation Systems (ITS), Network Intrusion Detection Systems (NIDS), Machine Learning, Deep Learning*

# ÖZET

## NESNELERİN İNTERNETİ TABANLI ARAÇ TİPİ SINIFLANDIRMA VE AĞ ANOMALİSİ TESPİTİ İÇİN MAKİNE ÖĞRENMESİ YAKLAŞIMLARI

Burak KOLUKISA
Elektrik ve Bilgisayar Mühendisliği Anabilim Dalı Doktora
Tez Danışmanı: Prof. Dr. V. Çağrı GÜNGÖR
Ocak 2024

Bu tez, Nesnelerin İnterneti kapsamında Akıllı Ulaşım Sistemleri ve Ağ Saldırı Tespit Sistemleri alanlarında yenilikçi yaklaşımlar sunmaktadır. Nesnelerin İnterneti teknolojilerinden yararlanılarak, Akıllı Ulaşım Sistemleri için düşük maliyetli, pil ile çalışan 3 boyutlu manyetik sensör geliştirilmiştir ve bu sensör araç tiplerinin sınıflandırılmasını sağlamaktadır. Araştırma, makine öğrenimi ve derin öğrenme modellerini, aşırı örnekleme, özellik seçimi ve çıkarma yöntemleri, hiperparametre optimizasyonu ve sinyallerin 2 boyutlu görüntülere dönüştürülmesi de dahil olmak üzere bir dizi teknikle geliştirmektedir. Araç tipi sınıflandırması için yeni yöntemler önerilmiş, bu yöntemler sınıflandırma performansını artırarak %92.92'ye varan bir doğruluk elde etmiştir. Ayrıca, bu tür uygulamalar için IoT cihazlarına artan bağımlılık, önemli siber güvenlik risklerini de beraberinde getirmektedir. Bu güvenlik açıklarını azaltmak için, ağ anomalisi tespiti için paralel bir yapay arı kolonisi (LR-ABC) algoritması ile eğitilmiş yeni bir lojistik regresyon modeli önerilmiştir. Bu model, tespit yeteneklerini geliştirmek için hiperparametre optimizasyonunu içermekte ve popüler NIDS veri kümelerinde %88.25 ve %90.11 doğruluk oranlarıyla üstün performans sergilemektedir. Genel olarak, bu araştırma, sağlam, ölçeklenebilir ve verimli çözümler sunarak Nesnelerin İnterneti ve Nesnelerin İnterneti siber güvenliğinde ilerlemeye katkıda bulunmaktadır. Bu yenilikler, sadece Nesnelerin İnterneti çağında araç tiplerinin sınıflandırmasını ve ağ güvenliğini artırmakla kalmayıp, giderek daha bağlantılı bir dijital manzarada gelecekteki IoT altyapısının gelişimine de öncülük etmektedir.

*Anahtar kelimeler: Nesnelerin İnterneti, Akıllı Ulaşım Sistemleri, Ağ Saldırı Tespit Sistemleri, Makine Öğrenmesi, Derin Öğrenme*

# Acknowledgements

First and foremost, I extend my deepest gratitude and sincere appreciation to my supervisor, Dr. V. Çağrı GÜNGÖR. His constant support and insightful guidance have been invaluable throughout my Ph.D. journey. He has not only been a mentor but also an inspiration in shaping me into a diligent researcher. I am truly fortunate to have been his doctoral student.

My cordial thanks also extend to Dr. Celal ÖZTÜRK, Dr. Özkan Ufuk NALBANTOĞLU, Dr. Abdulkadir KÖSE, and Dr. Rıfat KURBAN for serving on my dissertation defense committee.

Finally, my sincere appreciation goes to my parents and my wife. Their patience, continuous support, and encouragement have been a source of strength in my pursuit of this thesis.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| 2-D | Two-Dimensional |
| 3-D | Three-Dimensional |
| ABC | Artificial Bee Colony |
| ANOVA | Analysis of Variance |
| CE | Cross-Entropy |
| CNN | Convolutional Neural Networks |
| CPU | Central Processing Unit |
| CV | Cross-Validation |
| DL | Deep Learning |
| DNN | Deep Neural Network |
| DT | Decision Tree |
| FL | Focal Loss |
| FPR | False Positive Rate |
| FNR | False Negative Rate |
| GPU | Graphics Processing Unit |
| GRU | Gated Recurrent Unit |
| GS-CV | Grid Search Cross-Validation |
| IoT | Internet of Things |
| ITS | Intelligent Transportation Systems |
| KNN | K-Nearest Neighbor |
| LDA | Linear Discriminant Analysis |
| LR | Logistic Regression |
| LR-ABC | Logistic Regression trained by Artificial Bee Colony |
| LSTM | Long Short-Term Memory |
| ML | Machine Learning |
| MLP | Multi-Layer Perceptron |
| NIDS | Network Intrusion Detection Systems |
| OvO | One-vs-One |
| PRE | Precision |
| REC | Recall |

| | |
|---|---|
| ReLU | Rectified Linear Unit |
| RF | Random Forest |
| RNN | Recurrent Neural Network |
| SMOTE | Synthetic Minority Oversampling Technique |
| SVM | Support Vector Machine |
| XGBoost | Extreme Gradient Boosting |

*To my family*

# Chapter 1

# Introduction

In our rapidly evolving digital landscape, the Internet has become an indispensable part of modern life, enabling people to access communication, information, education, entertainment, and e-commerce in the easiest and fastest way possible. The Internet is a network that connects computers, smartphones, tablets, and many other electronic devices. With the development of smart devices using the Internet, the need for the Internet of Things (IoT) has increased. Along with the IoT, many devices, from our homes to cars, from industrial equipment to health devices, have become connected to the internet without human intervention [1]. IoT encompasses a wide range of applications and technologies where everyday objects and devices are connected to the internet, enabling them to collect and exchange data for various purposes. Figure 1.1 depicts a comprehensive illustration of the integrated architecture utilized within the scope of IoT systems. The advent of IoT has paved the way for more specialized applications, such as Intelligent Transportation Systems (ITS). ITS represents a convergence of information technology and transportation infrastructure, aimed at enhancing traffic efficiency and road safety [2].

ITSs are a specific application area within IoT that focuses on improving transportation and mobility through the use of connected sensors, devices, and data analytics. In the context of ITS, vehicle type classification using IoT technologies involves equipping vehicles with sensors and communication devices to gather data about their characteristics and behaviors. This data can include information about the type of vehicle, its speed, location, and more. In this thesis, Machine Learning (ML), and Deep Learning (DL) techniques are then applied to analyze this data and classify vehicles type based on the signals captured by the Three-Dimensional (3-D) magnetic sensors. Effective vehicle type classification is essential for improving traffic management and

congestion control, facilitating long-term infrastructure planning, enhancing public transportation and urban planning, ensuring environmental monitoring, and promoting road safety. These contributes significantly enhance the overall quality of urban life and form the backbone of modern ITS solutions.

**Figure 1.1 An illustration of an integrated architecture of Internet of Things [3].**

The IoT encompasses a vast array of internet-connected entities, from sensors and actuators to various smart devices, all generating immense volumes of data [4]. This wide spread of IoT devices, along with their growing complexity and important integration into infrastructures like ITS, makes them much more vulnerable to cyberattacks [5]. In this context, Network Intrusion Detection Systems (NIDS) emerge as indispensable to the security of IoT ecosystems. By providing comprehensive monitoring, robust protection against threats, ensuring data integrity, and compliance with regulatory mandates, NIDS are foundational to safeguarding IoT devices and their networks. Their role is not merely protective but pivotal for the successful deployment and operational integrity of IoT technologies. This thesis presents a machine learning approach to detect network anomalies and mitigate the multifaceted challenges posed by networks, such as high dimensionality, class imbalance, and the dynamic nature of network threats.

## 1.1 Research Objectives and Solutions

The IoT integrates a network of smart devices into our daily routines, establishing a complex array of data and connections that present both opportunities and challenges in terms of security and interpretation. This thesis recognizes the reality that each smart device becoming a part of the vast IoT network unleashes enormous potential for changing our lives, a potential that can only be realized with the development of advanced artificial intelligence algorithms and rigorous security mechanisms. Therefore, this thesis embarks on a pioneering exploration into vehicle type classification and network security within the IoT framework, employing ML and DL algorithms.

The classification of vehicle types is critical for the efficient functioning of urban traffic systems, the reduction of carbon emissions, and the improvement of transportation infrastructure, which is increasingly vital due to urban growth. ITS exemplifies the transformative impact of IoT in enhancing the efficiency and safety of movement within our cities and beyond. The application of sophisticated machine learning and deep learning techniques, when integrated with a single 3-D magnetic sensor and appropriate preprocessing methodologies, is posited to effectively address the issues of high dimensionality and class imbalance in vehicle datasets, consequently enhancing the precision and efficacy of vehicle type classification. Accordingly, novel approaches to vehicle type classification have been introduced in traffic management. However, the scope of this thesis extends beyond vehicle type classification; it seeks to innovate in the development of NIDS that serve as the sentinels of the IoT domain. As our reliance on IoT devices for privacy and security grows, the centrality of NIDS in protecting the integrity of our digital ecosystems becomes indisputable. This thesis tackles the challenges associated with ML-based NIDS, including the complexity of processing high-dimensional data, the persistent issue of class imbalance, and the elusive nature of network threats. It posits that integrating the machine learning model with the swarm intelligence algorithm will significantly enhance the efficacy of anomaly-based NIDS, surpassing the capabilities of traditional systems. Additionally, the efficiency of the proposed model is expected to benefit from parallel computing techniques, leading to quicker processing and response times. In general, it aims to develop a robust defense mechanism capable of not just detecting anomalies but also predicting and adapting to emerging threat patterns, thereby establishing a secure framework for the IoT.

Overall, the thesis begins with an in-depth evaluation of vehicle type classification, introducing two novel approaches. It then presents a novel approach for enhancing network anomaly detection on NIDS. The research primarily delves into three specific studies:

1. A deep neural network approach with hyper-parameter optimization for vehicle type classification using 3-D magnetic sensor [6].
2. Deep learning approaches for vehicle type classification with 3-D magnetic sensor [7].
3. An efficient network intrusion detection approach based on logistic regression model and parallel artificial bee colony algorithm [8].

### 1.1.1 A deep neural network approach with hyper-parameter optimization for vehicle type classification using 3-D magnetic sensor

This chapter endeavors to enhance vehicle type classification techniques within ITS by leveraging innovative sensor technology and advanced ML algorithms. The aim is to develop a system adept at distinguishing between different vehicle types using a novel methodology. The proposed systems include a 3-D magnetic sensor placed on a single-lane road to detect magnetic disturbances from vehicles. This system includes a sensor mote to capture sensor readings, a gateway for data transmission to the data center, and a web server for data processing and presentation.

Two datasets are generated: the signal data, derived from the magnetic sensor's raw signal and processed through zero-padding, and the extracted data, comprising 44 features extracted from the signal data. Both datasets undergo normalization for consistency. The raw signal data, while comprehensive, may contain excessive noise or irrelevant information that could mislead ML models, leading to high dimensionality, overfitting, and increased computational costs. Feature extraction and selection refine the data, reducing dimensionality and computational demands while enhancing model accuracy and interpretability. The extracted dataset, optimized with the best-selected features for each classifier, addresses these challenges.

The thesis employs several classification algorithms, including C4.5, Random Forest (RF), Logistic Regression (LR), Extreme Gradient Boosting (XGBoost), Support

Vector Machine (SVM), and Deep Neural Network (DNN), implemented using Python, Scikit-Learn, and Keras. The DNN, specifically, is designed with a unique architecture to enhance performance. It incorporates techniques like batch normalization and dropout rates to prevent overfitting and employs the Focal Loss (FL) function, which is pivotal in addressing class imbalance issues by focusing more on challenging cases. The methodology also includes rigorous testing and validation protocols. The datasets are split, with 70% used for training and 30% for testing. The training process involves 5-fold Cross-Validation (CV) to ensure the robustness and generalizability of the models. Hyperparameter optimization is carried out using a grid search optimization algorithm with CV, fine-tuning each model for best performance.

Overall, the findings indicate that feature extraction and selection enhance the performance of ML algorithms, while the FL function improves DNN results. Notably, the DNN, optimized with the selected 30 features using Grid Search Cross-Validation (GS-CV), achieves superior performance with an accuracy of 91.15% and an f-measure of 91.50%.

### 1.1.2 Deep learning approaches for vehicle type classification with 3-D magnetic sensor

In this chapter, a novel approach to vehicle type classification is explored by applying advanced DL techniques, including Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), and transfer learning algorithms such as VGG16, VGG19, Xception, MobilNet, MobilNetV2, DenseNet121, DenseNet169, and DenseNet201. Additionally, a custom soft voting ensemble model is employed to enhance performance results. A key innovation is the conversion of vehicle signal data into Two-Dimensional (2-D) image formats, enabling the use of sophisticated image-based DL models to improve classification effectiveness.

The dataset is carefully segmented, allocating 30% as the test set using stratified random sampling. The training set is enriched with the Synthetic Minority Oversampling Technique (SMOTE) to increase sample diversity. The core classification strategy involves LSTM and GRU models, which are well-suited for time-series data. This requires reshaping the dataset and applying masking due to different sample lengths. The transfer learning approach is also employed, where the vehicle signals are converted into

2-D images. This approach leverages pre-trained models, freezing their convolutional base layers and retraining the top layers. In addition to these advanced classification methods, the study also incorporates SVM for its simplicity and effectiveness in handling classification tasks. Hyperparameter optimization using grid search ensures that SVM classifiers are trained with the most suitable parameters.

Overall, the findings indicate that converting signal data into 2-D images enhances the performance of the vehicle type classification. The custom soft voting ensemble method, which combines time-series and image data for LSTM, GRU, and VGG16 models, demonstrates remarkable performance, achieving an accuracy of 92.92% and an f-measure of 93.42%. This marks an improvement over previous studies, especially in terms of accuracy and f-measure performance. The insights from this study can lead to the development of more reliable and precise classification systems, potentially integrating with an ensemble method to further improve classification performance.

### 1.1.3 An efficient network intrusion detection approach based on logistic regression model and parallel artificial bee colony algorithm

This chapter develops a sophisticated approach to detecting anomalies in networks, addressing the escalating cybersecurity challenges in the context of the exponential growth of the Internet and IoT. Recognizing the limitations of current NIDS, particularly in terms of accuracy, f1-measure, false positive rate, and false negative rate, the research introduces a novel anomaly-based detection methodology.

The core of this innovative approach is a new anomaly-based NIDS approach using LR, known for its straightforwardness, rapid classification in real-time, and efficiency. To circumvent LR's tendency to convergence to poor local minima, the system is developed with the Artificial Bee Colony (ABC) algorithm. This algorithm, inspired by the natural world, mimics the food-gathering patterns of honey bees and provides several advantages: (i) it requires minimal prior knowledge about the data and human intervention, allowing for classification without specific data preprocessing techniques; (ii) hybridizing the ABC approach with ML techniques improves model results; (iii) the ABC method is less dependent on known labels within the dataset compared to many ML approaches; (iv) it is inherently distributed and performs well in parallel and distributed computing environments.

The proposed model undergoes rigorous evaluation against ML and DL models on two publicly available NIDS datasets, UNSW-NB15 and NSL-KDD, each providing training and test sets. Essential data preprocessing steps, such as one-hot encoding and data normalization, are applied to ensure accurate and efficient processing of the categorical and numerical data within the NIDS datasets.

In summary, this chapter presents a comprehensive and efficient approach to NIDS using an LR trained by the ABC algorithm (LR-ABC). The proposed LR-ABC model stands out for its efficiency, particularly in reducing computational time through Central Processing Unit (CPU) and Graphics Processing Unit (GPU) parallelization techniques. The findings demonstrate the model's effectiveness, achieving a significant accuracy of 88.25% on the UNSW-NB15 dataset and 90.11% on the NSL-KDD dataset.

## 1.2 Research Outlines

The thesis is meticulously organized to ensure thoroughness and clarity. Chapter 2 details the specifications and functionalities of the 3-D magnetic sensor node. It then comprehensively explains ML and DNN classification algorithms. The chapter establishes a foundation by defining the evaluation metrics pivotal for measuring the algorithms' efficacy. A description of the employed dataset sets. The process of feature extraction is then systematically detailed, followed by a discourse on the ANOVA F-test feature selection method, which is instrumental in pinpointing key features. The chapter advances with a discussion on the adoption of the focal CE loss function, thoughtfully chosen to counteract class imbalance within the dataset. A thorough examination of grid search techniques follows, emphasizing their significance in hyperparameter optimization to enhance model performance. The chapter concludes with a presentation of the experimental procedures, the results obtained, and a comparative analysis of the proposed approach against existing classification algorithms.

Chapter 3 further expands the methodological discourse, starting with an introduction to the SMOTE for rectifying data imbalances. It introduces an innovative method for 2-D multi-color data visualization, thereby enriching the analysis with a new visual perspective. The chapter then provides an in-depth examination of recurrent neural network (RNN) paradigms, including LSTM and GRU, alongside transfer learning techniques. The narrative concludes with the integration of a custom soft voting ensemble

model, followed by a detailed presentation of experimental approaches and outcomes, and a comparative analysis that benchmarks the proposed model against traditional classification techniques. This chapter not only builds upon the foundational methodologies presented in Chapter 2 but also showcases the fusion of various advanced algorithms to create a robust vehicle classification system.

Chapter 4 provides a detailed description of the dataset used in NIDS and establishes a foundation by defining the crucial evaluation metrics for assessing algorithm efficacy. It explains the processes of normalization, one-hot encoding, and the implementation of Bayesian hyperparameter optimization to fine-tune the detection system. The chapter then introduces the fundamental concepts of LR and the ABC algorithm, laying the theoretical groundwork for the proposed model and addressing computation optimizations on CPUs and GPUs. Lastly, it presents a thorough analysis of the model's performance on publicly available NIDS datasets.

Finally, Chapter 5 summarizes the research findings, explains the societal impact and contribution to global sustainability, and outlines several potential areas for future investigation.

# Chapter 2

# A deep neural network approach with hyper parameter optimization for vehicle type classification using 3-D magnetic sensor

This chapter evaluates vehicle type classification using a single 3-D magnetic sensor and ML algorithms. It introduces an innovative ML approach, leveraging DNN with FL. This method incorporates hyperparameter optimization and feature extraction and selection techniques to effectively address the challenge of vehicle type classification. The proposed method is comprehensively investigated alongside other classification techniques. The proposed method undergoes comprehensive investigation and comparison with other classification techniques, providing a detailed analysis of its effectiveness.

## 2.1 Motivation and Related Work

According to the latest published data from the International Organization of Motor Vehicle Manufacturers, the total number of vehicles produced worldwide in 2019 was 92 million, and the current global vehicle population is approximately 1.32 billion [9]. The significant increase in the number of vehicles has led to various problems. In recent years, specific attention has been given to ITS to enhance the quality of life, improve traffic administration, and plan road maintenance effectively [10]. In ITS, traffic monitoring systems gather data, including the number, types, and speed of vehicles, to manage roadway systems, predict transportation needs, and enhance safety.

In many countries, significant investments are being made to develop, implement, and maintain traffic monitoring systems. For effective traffic planning, it is crucial to classify vehicle types correctly. Recently, researchers have studied different vehicle classification systems to accurately classify vehicle types. Due to significant technical challenges, various systems have been proposed using different technologies, including accelerometers [11, 12], acoustic sensors [13, 14], loop detectors [15, 16], LIDAR [17], piezoelectric sensors [18], vibration sensors [19], [20], magnetic sensors [21-30], cameras [31], and hybrid methods [32]. In Table 2.1, existing studies are compared and summarized based on different technologies.

**Table 2.1 A summary of research using different technologies for vehicle type classification.**

| Study | Technology | Sample Size | ACC (%) | Cost | Energy Efficiency |
|---|---|---|---|---|---|
| [3] | Accelerometer | 226 | 99.0 | NA | NA |
| [4] | Accelerometer | 142 | 89.0 | NA | NA |
| [5] | Acoustic | 160 | 73.42 | NA | NA |
| [6] | Acoustic | 106 | 71.69 | NA | NA |
| [24] | Hybrid | 50 | 90.0 | NA | NA |
| [7] | Loop Detectors | 1.330 | 94.21 | NA | NA |
| [8] | Loop Detectors | 21.600 | 91.0 | NA | NA |
| [10] | LIDAR | 872 | 86.90 | NA | NA |
| [11] | Vibration | 354 | 80.22 | NA | NA |
| [12] | Vibration | 415 | 89.41 | NA | NA |
| [13] | Magnetic | 5.837 | 88.0 | NA | NA |
| [14] | Magnetic | 188 | 83.0 | $50 | NA |
| [15] | Magnetic | 253 | 93.66 | NA | NA |
| [16] | Magnetic | 20.353 | 96.40 | NA | NA |
| [17] | Magnetic | 12.085 | 97.65 | $80 | NA |
| [18] | Magnetic | 1.442 | 80.55 | NA | NA |
| [19] | Magnetic | 300 | 95.46 | NA | NA |
| [20] | Magnetic | 732 | 95.40 | NA | NA |
| [21] | Magnetic | 412 | 94.41 | NA | NA |
| [22] | Magnetic | 6.042 | 97.83 | NA | NA |

Developing vehicle type classification systems poses extremely challenging tasks. Sensor types, hardware and parameter settings, configuration processes, operating environments, resistance to weather and noise, durability (battery life), and even

maintenance and installation costs are important features and requirements for these technologies. Compared to other technologies, magnetic sensors are particularly preferred due to their extreme climate resistance, compact size, easy set-up, and reasonable price. Additionally, given the effectiveness of the ML algorithms in various fields [33-37], researchers are exploring the classification of vehicle types using 3-D magnetic sensors based on ML. Several ML methods are applied, including Back Propagation Neural Networks [26, 27], SVM [25-28], RF [26-28], XGBoost [26], C4.5 [25-28], K-Nearest Neighbor (KNN) [25], [27], [28], Naive Bayes [25], and Convolutional Neural Networks (CNN) [30], for vehicle type classification using 3-D magnetic sensors.

Overall, it is evident that there is a need to improve the classification of vehicle types using ML algorithms. While a significant amount of research has been conducted using ML algorithms, a comprehensive analysis of these algorithms is essential. Therefore, this chapter efficiently addresses feature extraction, feature selection, and hyper-parameterization. Additionally, a new method is proposed, demonstrating improved results in vehicle type classification.

This chapter is organized as follows: Section 2.2 details the specifications and functionalities of the 3-D magnetic sensor node. Section 2.3 outlines the proposed approach and introduces the classification methods employed in this chapter. Section 2.4 is dedicated to the performance evaluation of the proposed approach, comparing it with various classification algorithms. The final section provides a comprehensive discussion of the results obtained in this chapter.

## 2.2 Magnetic Sensor

The proposed system consists of a 3-D magnetic sensor for measuring the intensity of magnetic fields, a mote responsible for reading the sensor outputs, a gateway responsible for transmitting the data provided by the sensor mote to the data center, and a web server responsible for analyzing and displaying the collected data. The capsule structure that protects the system from environmental factors is shown in Figure 2.1. The proposed system diagram is displayed in Figure 2.2.

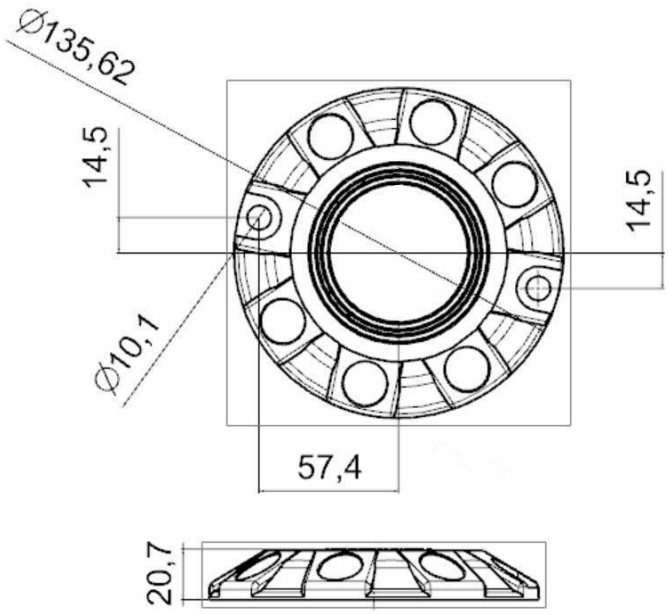


**Figure 2.1 Illustration of the node's size and shape in the three-dimensional magnetic sensor system.**

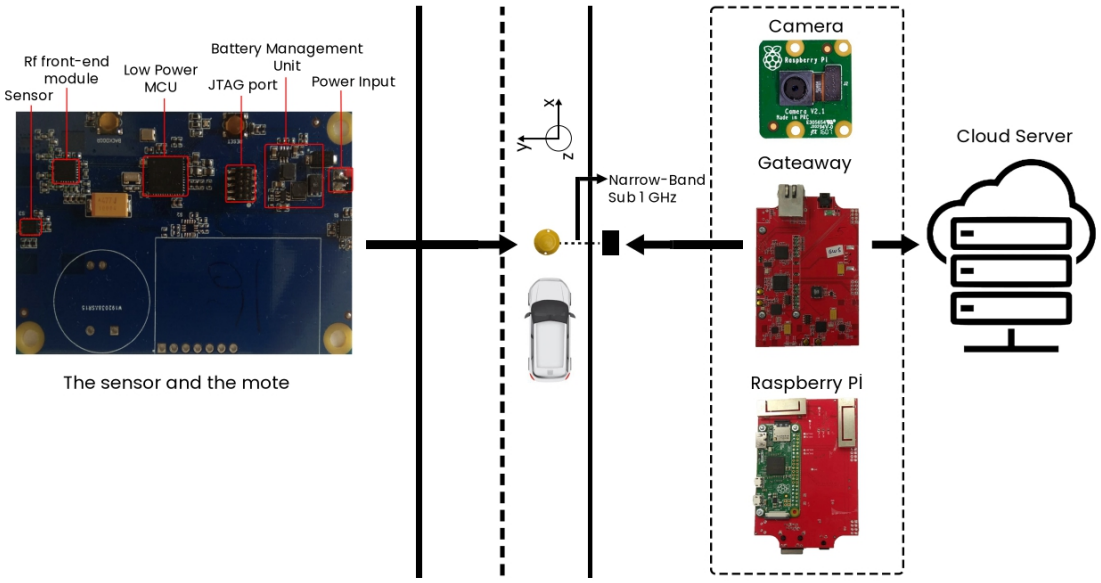


**Figure 2.2 The illustration diagram of the proposed system using the three-dimensional magnetic sensor for vehicle type classification.**

### 2.2.1 Sensor

In order to extract new features for the classification algorithms, a single 3-D magnetic sensor is used as a sensing unit. The sensor node's features are as follows: The supply voltage ranges from 0.9 to 3.6 volts (V), the maximum data rate is 5 kilobits per second (kbps), the transmit (TX) power is +30 decibels relative to one milliwatt (dBm), the radio frequency (RF) ranges from 863 to 876 megahertz (MHz), the RF communication distance ranges from 100 to 700 meters (m), the operating temperature ranges from -25 degrees Celsius (°C) to 80°C, and the mechanical robustness is up to 10,000 kilograms (kg). It has a waterproof ingress protection rating of IP67, signifying its robust defense capabilities. The digit '6' refers to complete protection against solid matter like dust, and the digit '7' refers to its ability to resist liquid intrusion (immersion up to 1 meter). The battery's lifetime is 2 years. The system architecture of a 3-D magnetic sensor node is shown in Figure 2.3.

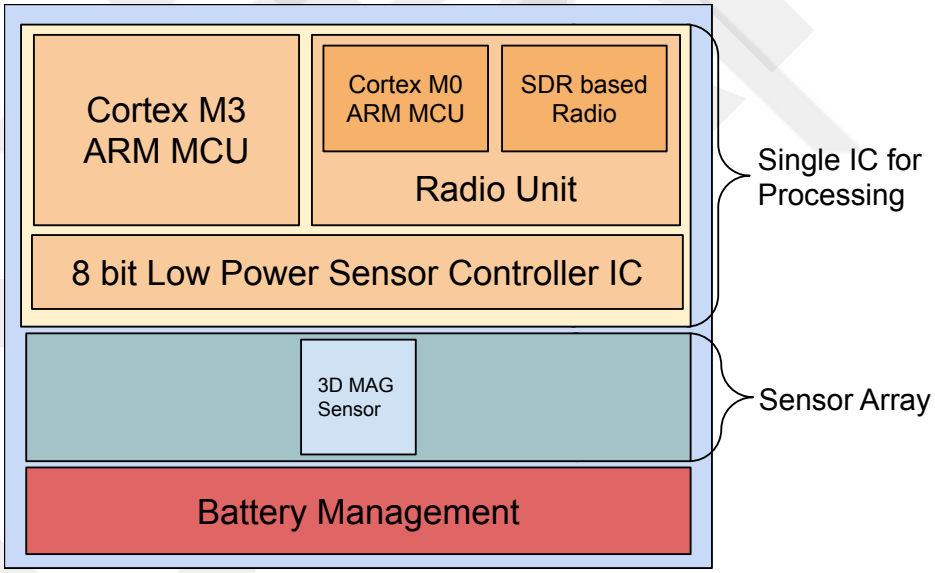


**Figure 2.3 The system architecture of a three-dimensional magnetic sensor node.**

The cost analysis of the mote is as follows: The microcontroller unit (MCU, model CC1312) costs $7, the RF module (model SE2435L) costs $3, the magnetic sensor array costs $5, the flash memory costs $1, the antenna costs $2, and the printed circuit board (PCB) costs $7. The features of sensor nodes are shown in Table 2.2, and the cost of each component is provided in Table 2.3. The proposed sensor node has a total cost of $25,

which is not mentioned in most of the studies. A limited number of recent studies [19], [22] mentioned that their sensor costs are $50 and $80, respectively, while our sensor node costs much less compared to these studies.

**Table 2.2 The sensor node's features.**

| Feature | Description |
|---|---|
| Voltage | 0.9-3.6 V |
| Max. Date rate | 5 kbps |
| Tx Power | +30 dBm |
| RF Frequency | 863–876 MHz |
| RF Communication distance | 100 m–700 m |
| Operation temperature | −25 +80 ◦C |
| Mechanical robustness | 10 000 kg |
| Waterproof IP rating | IP 67 |
| Batter lifetime | 2 years |

**Table 2.3 Detailed cost analysis of the mote in the three-dimensional magnetic sensor system.**

| Component part | Price |
|---|---|
| MCU (CC1312) | $7 |
| RF Module (SE2435L) | $3 |
| Magnetic sensor array | $5 |
| Flash memory | $1 |
| Antenna | $2 |
| PCB | $7 |
| Total | $25 |

### 2.2.2 Mote

The developed mote includes a CC1312 wireless MCU, which features a 48-MHz Cortex-M4F microcontroller, a special radio controller based on Cortex-M0, an ultralow-power 8-bit sensor controller integrated circuit (IC), 80 kilobytes (kB) of static random access memory (SRAM), a universal asynchronous receiver/transmitter (UART), an inter-integrated circuit (I2C), and a serial peripheral interface (SPI) [38]. Communication with the gateway is ensured by the CC1312 wireless MCU operating at the 868 MHz

band, adopting the low-rate wireless personal area network (LR-WPAN), which is an Institute of Electrical and Electronics Engineers (IEEE) 802.15.4 communication protocol. An RF front-end module is adopted to increase the RF power output [39]. The developed embedded software on the mote is used for sensor calibration, real-time sensor measurement reading, temporary data processing and storage, and finally forwarding the stored data to the gateway. A picture of the sensor and the mote is provided with labeled components in Figure 2.4.

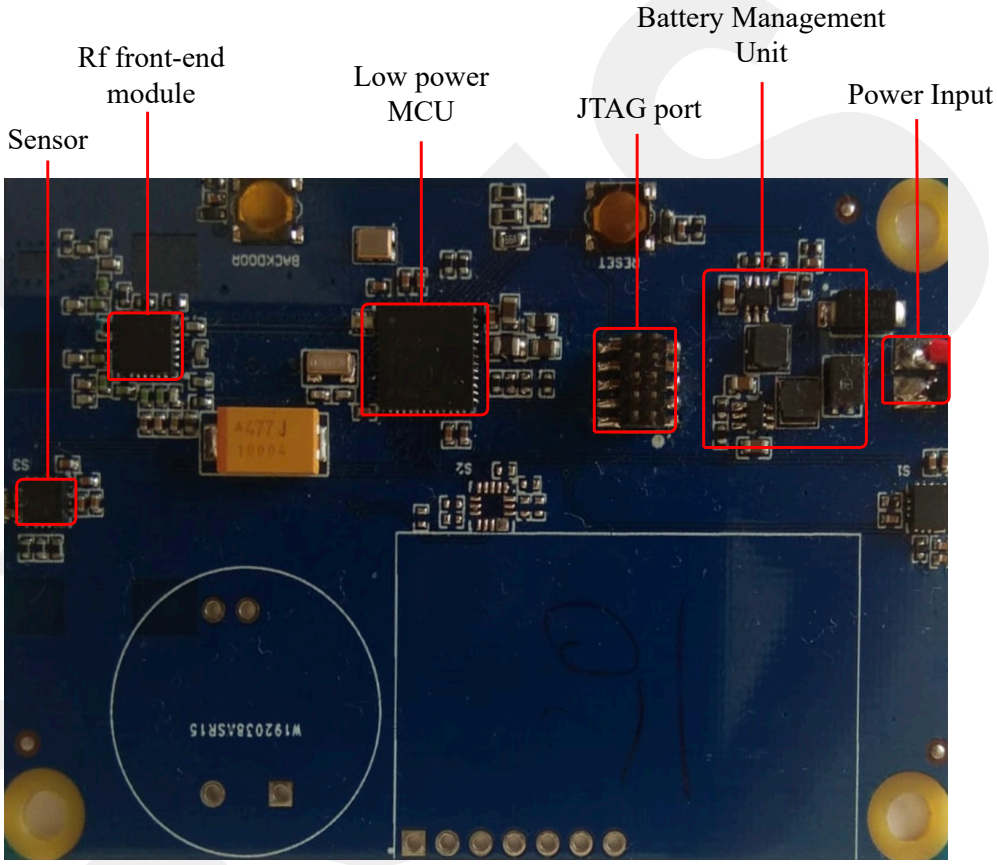


**Figure 2.4 The sensor unit and the mote within the three-dimensional magnetic sensor system.**

### 2.2.3 Gateway

The gateway is equipped with the same mote hardware connected to a Raspberry Pi [40]. The data acquired from sensor motes is transmitted to the data center via the Linux operating system, which is run on the Raspberry Pi. The same type of transceiver is used for the mote connection. The gateway features are as follows: The voltage supply is 5 V; the TX power is +30 dBm; the RF frequency is between 863 and 876 MHz; the

RF communication distance ranges from 100 to 700 m; the operation temperature range is between −25 °C and 80 °C; and the maximum number of connected nodes can be up to 100. The gateway properties are provided in Table 2.4.

**Table 2.4 Technical Specifications of the Gateway.**

| Feature | Description |
|---|---|
| Voltage | 5 V |
| Tx Power | +30 dBm |
| RF Frequency | 863–876 MHz |
| RF Communication distance | 100 m–700 m |
| Operation temperature | −25 +80 ◦C |
| Max. Number of connected nodes | 100 |

### 2.2.4 Backend

The sensor values are stored along with a reference camera system to facilitate data tagging and store the data for later offline processing. A backend cloud system has been developed. By synchronizing the camera with the magnetic sensor node, the user selects the time frame for cropping and tagging the videos with the magnetic sensor node's data. The video dataset is saved in the NoSQL database for future usage and long-term retention.

### 2.2.5 Battery lifetime

Two different measurement methods are used to obtain the magnetic signature of the vehicle. The first of these methods is the time-dependent measurement method, which is continuously based on receiving data from sensors at certain time intervals. The primary advantage of this method is energy consumption. However, there is a possibility that the sensor cannot detect significant magnetic field changes, leading to an increase in the error rate in the magnetic signature.

The second method, called vector magnitude-dependent measurement, is the method that takes samples according to certain magnetic field changes (e.g., every 10 microteslas). The primary benefit of this method is that it captures all magnetic field changes, allowing for a larger sample amount that can be processed and a more accurate

signature. However, adding timestamp information increases the size of the data to be transmitted in a narrow band, leading to a significant increase in battery consumption.

In light of this information, during the project's implementation, the second method was used to detect magnetic changes during the vehicle's entry and exit from the sensor. The first method was used to sample the vehicle's movement on the sensor based on time. Power consumption is optimized in this way.

As the vehicle passes over a 3-D magnetic sensor node, the node wakes up, records the vehicle's measurements, transmits them to the gateway, and then sleeps again. The sampling frequency of the sensor is set to 400 Hz, which is the maximum frequency, in order to obtain better signal data and detect the vehicle's passing with minimal time loss. Reducing the sampling frequency would decrease the current consumption of the sensor, thereby increasing the battery life of the sensor node. However, it would also result in a decrease in data quality, adversely affecting the performance results of vehicle classification.

The sensor node was mounted on a single-lane road. Magnetic distortions were measured for a total of 50 different vehicles, and the threshold (T) and references for X, Y, and Z were determined; the equation is shown in (2.1). For new vehicles, magnetic distortions are denoted as $Xo$, $Yo$, and $Zo$, while the reference magnetic distortions for the vehicles are $Xr$, $Yr$, and $Zr$. If the result of the equation is greater than the T value, the vehicle's magnetic distortions are recorded. The magnetic distortions of the 50 different vehicles were used to determine that T is equal to 110.

$$\sqrt[2]{(X_0 - X_r)^2 + (Y_0 - Y_r)^2 + (Z_0 - Z_r)^2} > \mathrm{T} \qquad (2.1)$$

The sensor consumes 20 microamperes (uA) during the sleep period. The average power consumption for a vehicle includes the sensor measurement period and packet transmission, with durations of 8 milliseconds (ms) and 12 ms, respectively. Packet transmission consists of pre-processing (10 milliamperes (mA) per ms, 2 ms), RX (1 amperes (A) per ms, 5 ms), TX (20 mA per ms, 3 ms), and post-processing (10 mA per ms, 2 ms). The sensor measurement period uses 40 mA per ms, and packet transmission

uses 425 mA per ms. In total, the average power consumption for a vehicle is 5420 mA with a duration of 20 ms. A vehicle's average power consumption is shown in Table 2.5.

**Table 2.5 Power consumption profile for vehicle sensing and sensor node operations.**

| Name | Average current (per ms) | Duration |
|---|---|---|
| Sensor measurement | 40 mA | 8 ms |
| Packets transmit (Pre-processing) | 10 mA | 2 ms |
| Packets transmit (RX) | 1 A | 5 ms |
| Packets transmit (TX) | 20 mA | 3 ms |
| Packets transmit (Post-processing) | 10 mA | 2 ms |

## 2.3 Classification Methods

This chapter presents a summary of the ML and DL techniques utilized in the development of this thesis. The examination begins by considering Decision Trees (DTs), with a particular focus on the C4.5 algorithm, known for its ability to process diverse data types and generate clear decision rules. The C4.5 algorithm lays the groundwork for advanced ensemble methods, such as RF and XGBoost, which enhance prediction accuracy by combining the strengths of multiple models, thereby reducing individual biases and increasing the overall model precision. In addition, the study examines LR, which is well-known for its unique ability to do binary classification and its flexibility in handling multiclass settings. SVM is also examined for its distinctive capability to identify the best hyperplane, a crucial element in both binary and multiclass classification applications. In the exploration of DNNs, the thesis uncovers their potential to reveal complex patterns hidden within high-dimensional data, outperforming the more limited linear models. The ensuing sections offer an in-depth review of these algorithms, examining their theoretical foundations, practical implementations, and rigorous empirical testing. The aim is to elucidate their strengths, pinpoint their limitations, and demonstrate their capacity to turn raw data into insightful analyses.

### 2.3.1 Decision trees (C4.5)

Decision trees, specifically the C4.5 algorithm, are a type of supervised learning method commonly employed in classification tasks. They are capable of handling both categorical and continuous input and output variables. The goal is to construct a model capable of predicting the value of the dependent variable by deriving straightforward decision rules from the independent variables. Entropy (E) and Gini (G) are used to determine these criteria in establishing these rules.

Entropy measures the uncertainty or unpredictability within a system, assessing the level of randomness in the class distribution among the dataset's examples. In a dataset with multiple classes, entropy for a dataset S can be calculated using equation (2.2), where $p_i$ represents the proportion of samples belonging to class $i$ within the dataset. Information Gain (IG) quantifies the difference in entropy between a dataset before and after it is divided based on a particular feature. It is used to decide the characteristic on which to divide at each stage of constructing the tree. IG is determined by subtracting the entropy before the split from the weighted sum of the entropies of each subgroup resulting from the split, as shown in equation (2.3). The aim is to identify the feature that yields the most significant decrease in entropy (the highest IG) for the split. $T$ represents the set of all possible outcomes (subsets) after the split. $S_t$ is the subset of $S$ corresponding to the outcome $t$, and $|S_t|\ /S$ is the weight of the subset $S_t$. By selecting splits that maximize IG, the model incrementally increases the predictability of the outcome, leading to a structured decision-making process that accurately captures the data's inherent patterns. The resultant prediction model takes the form of a tree, where each path from the root to a leaf represents a series of decisions culminating in a predicted outcome. Ideally, these paths clearly and accurately categorize the input samples based on their attributes.

$$E(S) = -\sum_{i=1}^{n} p_i \log_2(p_i) \tag{2.2}$$

$$IG(A,S) = E(S) - \sum_{t \in T} \frac{|S_t|}{S} E(S_t) \tag{2.3}$$

$$G(S) = 1 - \sum_{i=1}^{n} p_i^2 \tag{2.4}$$

The Gini index evaluates the probability of incorrect labeling of an element in the dataset if it were randomly labeled according to the distribution of labels in the subset. The Gini impurity of a dataset $S$ may be represented quantitatively using equation (2.4), where $p_i$ is the probability of an item being categorized into a class $i$ in the dataset $S$, and $n$ represents the total number of classes. The probability $p_i$ is calculated by dividing the number of things classified with class $i$ by the total number of items in the dataset. To compute the impurity, one subtracts the total of the squared probabilities of all classes from one. The Gini index is often favored in DT algorithms because it is computationally simpler than entropy. Unlike entropy, Gini does not need logarithmic computations, which are needed to compute the logarithm of probabilities. Despite their different mathematical formulations, Gini impurity and entropy often lead to the creation of similar DTs. However, due to its computational efficiency—particularly with large datasets and numerous features—the Gini index may be preferred in several implementations.

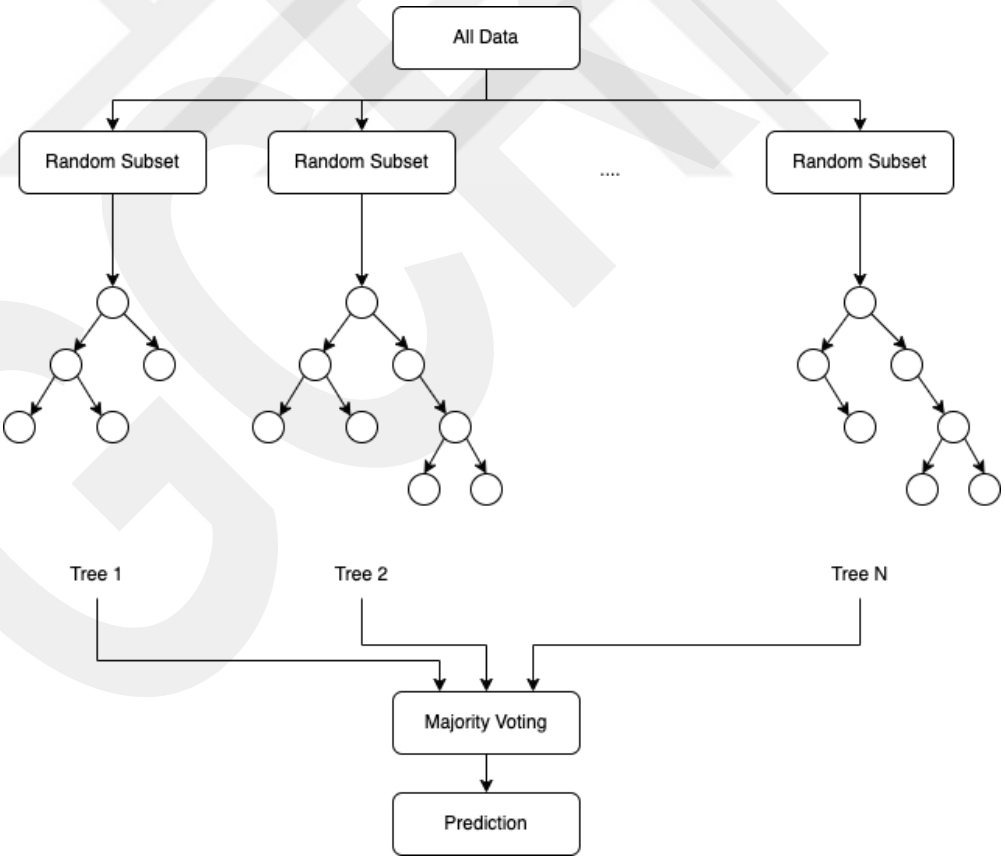


**Figure 2.5 Schematic representation of random forest classification model.**

### 2.3.2 Random forest (RF)

The Random Forest (RF) algorithm is an ensemble approach that constructs multiple DTs during the training phase and determines the class through majority voting based on the classifications provided by the individual trees [41]. The inherent nature of RF enables it to enhance the predictive accuracy of individual DTs by aggregating their results. By averaging out the biases of individual trees, the overall variance of the final model is reduced, resulting in a more precise and robust classifier. Moreover, RF handling overfitting effectively mitigates overfitting through the use of several trees in the ensemble. The process begins with bootstrapping the dataset, which involves generating numerous subsets from the original dataset using random sampling with replacement. Each subset is then used to train an individual DT. In constructing a DT, unlike standard DTs, each node is split using the best among a randomly chosen subset of predictors at that node. This introduces randomness into the model, ensuring a diverse collection of trees in the forest. Upon training completion for all the trees, predictions for new and unseen data are made by aggregating the votes from each tree to determine the most probable class. The final prediction of the RF is made by selecting the class with the highest number of votes from all the trees. Figure 2.5 depicts the schematic representation of RF classification.

### 2.3.3 Extreme gradient boosting (XGBoost)

Extreme Gradient Boosting (XGBoost) is an ensemble learning method [42], which means that it aggregates the predictions from multiple models to produce a final outcome. Unlike bagging, as utilized in RF, XGBoost constructs trees sequentially. Each successive tree is built to correct the errors of its predecessors. The term 'gradient boosting' refers to the method's ability to minimize a loss function by successively adding weak learners, typically DTs with limited depth, in a manner akin to gradient descent optimization. Weak learners in XGBoost are characterized as shallow trees with few levels. Despite their simplicity, when combined, these weak learners produce a powerful model. XGBoost builds upon the basic gradient boosting framework by introducing a more regularized model formulation, which helps to prevent overfitting and consequently leads to enhanced performance. During each iteration, XGBoost adds a new tree designed to predict the residuals or gradients of the cumulative model concerning the loss function. Furthermore, XGBoost implements L1 (Lasso Regression) and L2 (Ridge Regression) regularization

methods. These methods serve to constrain the magnitude of the feature weights, thus reducing the risk of overfitting and improving the model's ability to generalize to unseen data.

The objective function of XGBoost is a combination of a loss function and a regularization term. The aim is to minimize objective function as shown in equation (2.5), where $\theta$ represents the parameters of the model and $n$ is the number of training instances. The loss function $l\,(y_i, \hat{y}_i)$ quantifies the discrepancy between the predicted value $y_i$ and the actual label $y_i$. K is the number of trees and $\Omega(f_k)$ signifies the regularization term for the k-th tree. The regularization term $\Omega(f_k)$, which penalizes the model's complexity, is defined in equation (2.4), where γ controls the complexity through the number of leaves in the tree. $T_k$ is the count of leaves in the k-th tree, $\lambda$ is the L2 regularization term affecting the weights, and $w_k$ is the vector of leaf weights for the k-th tree.

$$Obj(\theta) = \sum_{i=1}^{n} l\,(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \tag{2.5}$$

$$\Omega(f_k) = \gamma T_k + \frac{1}{2}\lambda \|w_k\|^2 \tag{2.6}$$

At each iteration $t$, a new tree $f_t$ is added to the model to correct the errors made by the existing ensemble of trees. This new tree is trained to predict the residual errors of the model up to that point. The residual for the $i$-th instance following the $(t-1)$-th iteration is depicted in equation (2.7). Unlike the actual labels, $f_t$ is trained on these residuals. This approach allows the new tree to refine the predictions made by the prior ensemble. Once a tree is learned, the model undergoes an update, as specified in equation (2.8), with $\eta$ being the learning rate that modulates the rate at which the model adapts. This iterative process is repeated a set number of times or until the model converges. The final prediction for any instance is the aggregate of the predictions from all trees, as presented in equation (2.9).

$$r_i^{(t)} = y_i - \hat{y}_i^{(t-1)} \tag{2.7}$$

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + \eta\, f_t(x_i) \tag{2.8}$$

$$\hat{y}_i = \sum_{t=1}^{T} f_t(x_i) \tag{2.9}$$

### 2.3.4 Logistic regression (LR)

Logistic regression (LR) is a fundamental statistical technique widely used for binary and multi-class classification tasks [43] . It is very efficient in situations when the goal is to classify data points into one of two distinct classes based on a set of input features. This approach is based on the concepts of probability and works by evaluating the chance that a given instance belongs to a certain class. The logistic function, commonly referred to as the sigmoid function, plays a crucial role in LR, as seen in equation (2.10). In this equation, z represents a linear combination of the input features, where $B$ is bias and $w_1, w_2, \dots, w_n$ are coefficients for the inputs $x_1, x_2, \dots, x_n$ as illustrated in equation (2.11). The model parameters, $w_1, w_2, \dots, w_n$ are estimated using a training dataset. This is achieved by maximizing the probability of the observed data, often by minimizing a cost function such as the CE loss. The LR cost function is defined by equation (2.12), which $m$ is the number of training samples, $y^{(i)}$ represents the observed class label for the $i-th$ samples and $\sigma(z^{(i)})$ is the predicted probability that $y^{(i)} = 1$. Gradient descent techniques are used to determine the parameters $w$ that minimize the cost function $J(\beta)$. During the process of gradient descent, the parameters $w$ are updated in an iterative manner, moving in the direction that results in the fastest drop of the cost function. After the model has been trained, it is possible to make predictions on fresh data by using the logistic function. Figure 2.6 displays the schematic diagram for LR classification.

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.10}$$

$$z = B + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n \tag{2.11}$$

$$J(\beta) = -\frac{1}{m} \sum_{i=1}^{m} \left[ y^{(i)} \log \sigma(z^{(i)}) + (1 - y^{(i)}) \log(1 - \sigma(z^{(i)})) \right] \tag{2.12}$$

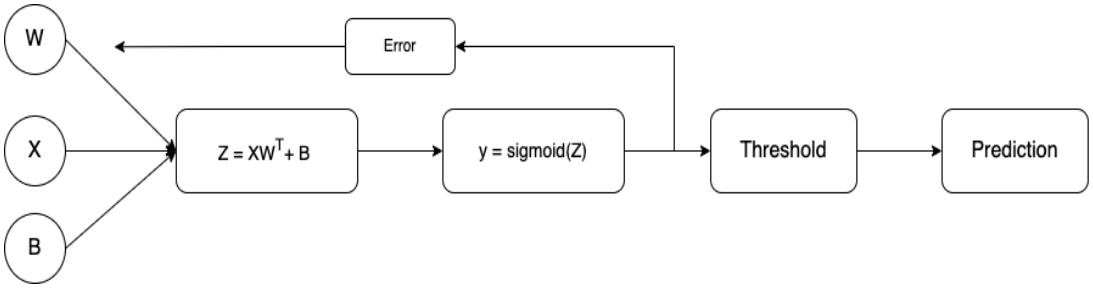


**Figure 2.6 Schematic diagram for logistic regression classification model.**

### 2.3.5 Support vector machine (SVM)

A Support Vector Machine (SVM) is a robust and flexible supervised ML algorithm utilized for the purposes of classification and regression. In binary classification, the core objective of SVM is to identify the optimal hyperplane that effectively separates the data points of one class from another. The hyperplane is selected to maximize the margin, defined as the distance between the hyperplane and the nearest data point of any class. For multiclass classification, the One-vs-One (OvO) approach is employed, wherein each class is compared against every other class in a series of binary classification tasks. A binary classifier is thus developed to predict the probability of a data item belonging to one class as opposed to another. For each pair of classes, a dedicated binary classifier is trained, and the class that predominates across all pairings is deemed the final classification. In the OvO context, the decision function for a SVM is derived from the principle of maximizing the margin between classes. This is usually expressed in equation (2.13), where the weight vector $w$ is orthogonal to the hyperplane and $x$ represents the input feature vector. The bias term, denoted as $b$, is responsible for offsetting the decision border from the origin. Each binary classifier aims to determine the most discriminative hyperplane that separates the two classes, as exemplified in Figure 2.7.

This is accomplished by solving the optimization problem presented in equation (2.14), while adhering to the constraints outlined in equation (2.15). Notably, the function $\phi(x_i)$ has the potential to map the input data to a space with a higher number of dimensions. The label $y_i$ represents the label assigned to the instance $x_i$, where $y_i$ belongs to the set {-1,1}. The slack variables, $\xi_i$, allow for degree of misclassification within the soft margin framework. The penalty parameter C optimizes the balance between the margin width and the classification error. In OvO SVM approach, the kernel function is used to convert the data into a space with greater dimensions, facilitating its separation

with a linear decision boundary. Subsequently, the decision function is used to effectively segregate the data inside this newly created domain.

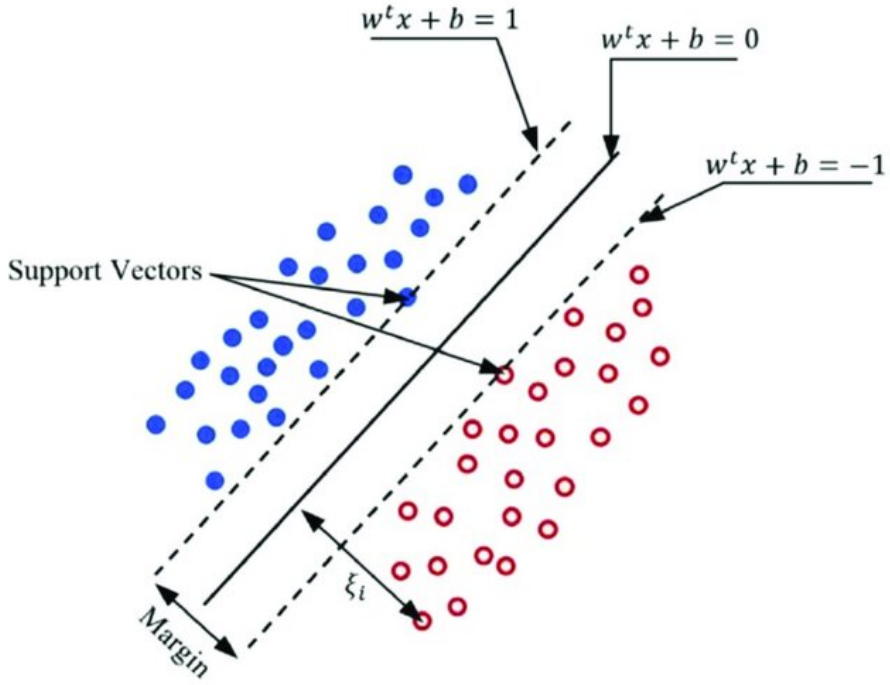


**Figure 2.7 Hyperplane illustration of support vector machine classification model [44].**

The kernel trick permits the SVM to learn a nonlinear decision boundary by implicitly transforms the input features into a high-dimensional space and calculating the inner products between the transformed data representation. Thus, relationships between pairs are computed without the need for explicit coordinates in the expanded space. The kernel function is defined as shown in the equation (2.16). After the SVM has undergone training, a voting technique is used for the purpose of multiclass classification. Each binary classifier provides a prediction for a novel input vector $x$. The class that receives the highest number of votes from the K (K - 1) / 2 classifiers is selected as the final prediction.

$$f(x) = sign(w^T x + b) \tag{2.13}$$

$$min_{w,b,\xi} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \xi_i \tag{2.14}$$

$$\begin{cases} y_i\ (w^T \phi(x_i) + b) \geq 1 - \xi_i, \\ \xi_i\ \geq 0 \\ i = 1, \ldots, n \end{cases} \tag{2.15}$$

$$K(x_i, x_j) = \phi(x_i)^T\ \phi(x_j) \tag{2.16}$$

### 2.3.6 Deep neural network (DNN)

Deep Neural Network (DNN) is a type of classification model that effectively discriminates between classes due to its non-linear capabilities. In a DNN, each neuron holds a single value known as the neuron's activation. Each layer contains multiple neurons, with the neurons in one layer connected to those in the subsequent layer through a network. Figure 2.8 illustrates an example of a DNN-based classification model. This model's architecture includes a single input layer, two hidden layers, and one output layer.

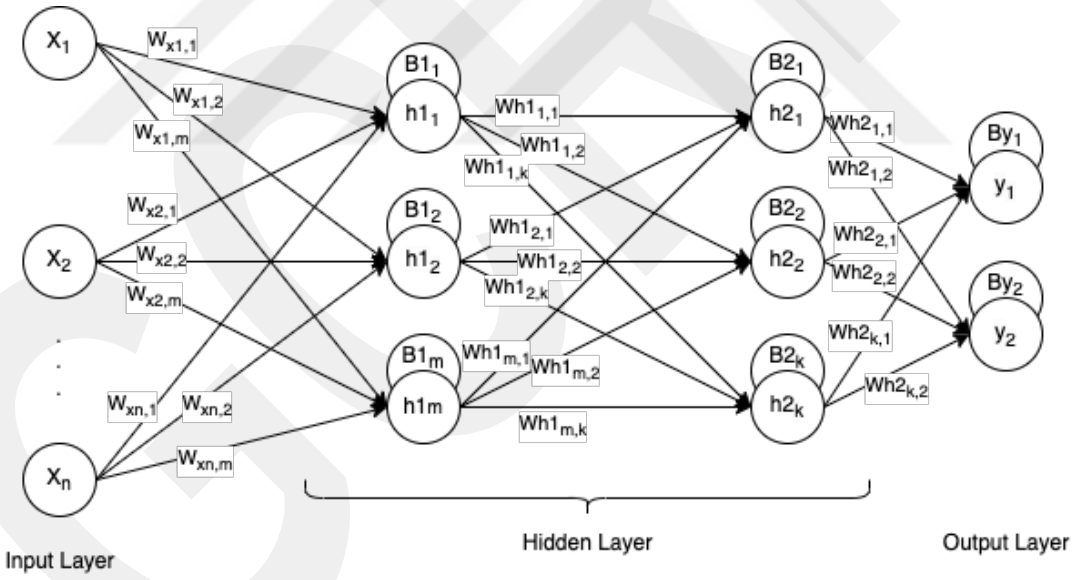


**Figure 2.8 Illustration of a deep neural network classification model.**

The computation for the first neuron in the first hidden layer is detailed and illustrated in equation (2.17). The Rectified Linear Unit (ReLU) functions as the activation mechanism within the hidden layers, and its mathematical formulation is provided in equation (2.18). Subsequently, the computation for the first neuron in the

output layer is described in equation (2.19), where the sigmoid function is applied as the activation function, as indicated in equation (2.20). The neurons $y_1$and $y_2$ represent the model's outputs. The closeness of these outputs to the actual values is evaluated using the binary CE loss function, which is presented in equation (2.21). The optimizer leverages this metric to incrementally adjust the model's weights, thereby enhancing the training process via iterative optimization.

$$h1_1 = ReLU(x_1 * w_{x1,1} + x_2 * w_{x2,1} + \ldots + x_n * w_{n1,1} + B1_1) \quad (2.17)$$

$$ReLU(x) = max(0, x) \quad (2.18)$$

$$y_1 = Sigmoid(h2_1 * wh2_{1,1} + h2_2 * wh2_{2,1} + \ldots + h2_k * wh2_{k,1} + By_1) \quad (2.19)$$

$$Sigmoid(x) = \frac{1}{1 + e^{(-X)}} \quad (2.20)$$

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i * log(p(y_i)) + (1 - y_i) * log(1 - p(y_i)) \quad (2.21)$$

The number of neurons in the input layer $n$ is variable and depends on the size of the dataset in use. While this example demonstrates a network with two hidden layers, the number of hidden layers can vary according to the complexity of the problem. In this instance, the network's output layer features two neurons, designed to distinguish between two class patterns. Should the research require a more nuanced classification (identifying different types of classes), parameters including the number of neurons in the output layer, the loss function, and the activation function of the output layer may be adjusted accordingly.

## 2.4 Vehicle Type Classification

### 2.4.1 Evaluation metrics

Accuracy is an essential criterion for evaluating a model's overall performance. The major goal of the current research is to increase accuracy, but the accuracy criterion may not be adequate in unbalanced datasets. Therefore, in addition to the accuracy metric, F1-measure, Precision (PRE), and Recall (REC) are also used to evaluate the classification performance. Precision reflects the proportion of correctly identified positive cases among all cases classified as positive. Recall (or sensitivity) measures the proportion of actual positive cases that are correctly identified. The F1-measure, the harmonic mean of recall and precision, reflects the model's sensitivity and robustness. These are important details to be examined in this study. These performance metrics are given in equations (2.22), (2.23), (2.24), and (2.25), respectively. These metrics help to assess the performance of the model in several aspects. The traditional confusion matrix is shown in Table 2.6.

**Table 2.6 Traditional confusion matrix.**

| | **Predicted Anormal** | **Predicted Normal** |
|---|---|---|
| **Actual Abnormal** | TP | FN |
| **Actual Normal** | FP | TN |

$$Accuracy\ (ACC) = \frac{TP + TN}{TP + FN + FP + FN} \tag{2.22}$$

$$\text{Precision (PRE)} = \frac{TP}{TP + FP} \tag{2.23}$$

$$Recall\ (\text{REC}) = \frac{TP}{TP + FN} \tag{2.24}$$

$$F1 - measure\ (F1) = \frac{2 * TP}{2 * TP + FP + FN} \tag{2.25}$$

### 2.4.2 Dataset

A 3-D magnetic sensor is built and mounted on a single-lane road. As vehicles pass over the sensor, it records magnetic disturbances caused by the metal in the vehicles, leading to varying signal changes. These distortions are primarily influenced by the vehicle's speed, physical characteristics, and environmental factors. Vehicles are classified into three groups according to their structure: light (L, including motorcycles as class 1), medium (M, encompassing passenger cars as class 2), and heavy (H, covering buses as class 3). In this classification, there are 376 labeled vehicle examples, with 46 light vehicles, 298 medium vehicles, and 32 heavy vehicles. Representative samples are shown in Figure 2.9. 'T' and 'Gauss' refer to milliseconds (ms) and Gauss (Gs) units, respectively. In each figure, the millisecond value varies according to the vehicle's impact on the sensor node. The *X*-axis values are negative, reflecting the sensor node's placement on the mote. The sensor records raw data from the X, Y, and Z axes as vehicles pass, with signal durations ranging from 13 to 207 ms. To standardize sample lengths, signals are zero-padded to a maximum duration of 207 ms. Considering the three axes in the first dataset (signal data), there are 621 features per sample. From the raw signal data, 44 features are extracted for the second dataset (extracted data).

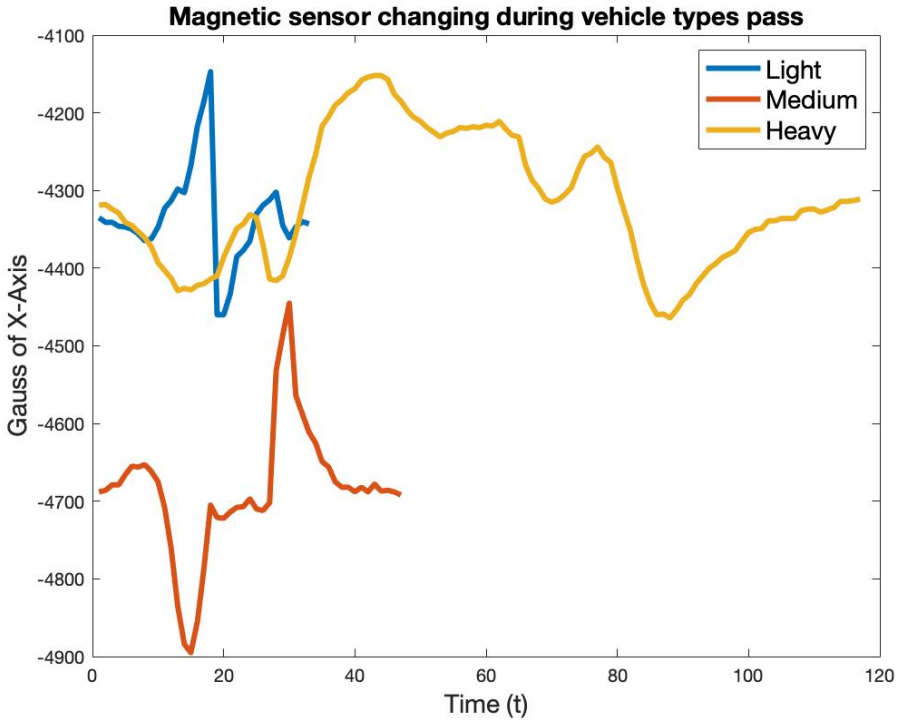


**Figure 2.9 Signal patterns for Light, Medium, and Heavy vehicles as captured by the three-dimensional magnetic sensor.**

### 2.4.3 Feature extraction

Excess data may complicate the classification process and lead to incorrect results. To reduce feature size and obtain better performance results, feature extraction techniques are employed. This process focuses on capturing the subtle changes in magnetic field strength as vehicles traverse a 3-D magnetic sensor. These variations are encapsulated in a set of features, each contributing uniquely to the vehicle classification profile. As shown in Table 2.7, a total of 44 features are extracted. A total of 44 distinct features are extracted to encapsulate these variations, as outlined in Table 2.7. Each feature is considered within the three axes—X, Y, and Z. For instance, maximum values are extracted for each axis, culminating in three separate features. The signal length remains constant across all axes, resulting in a single feature.

**Table 2.7 List of extracted features from three-dimensional vehicle signal.**

| No | Features | # |
|---|---|---|
| 1 | Maximum values (x, y, z) | 3 |
| 2 | Index of maximum (x, y, z) | 3 |
| 3 | Minimum values (x, y, z) | 3 |
| 4 | Index of minimum (x, y, z) | 3 |
| 5 | Length of signal (l) | 1 |
| 6 | Mean of the signals (x, y, z) | 3 |
| 7 | Median of the signals (x, y, z) | 3 |
| 8 | # of local maximum (x, y, z) | 3 |
| 9 | # of local minimum (x, y, z) | 3 |
| 10 | Mean of local maximum (x, y, z) | 3 |
| 11 | Mean of local minimum (x, y, z) | 3 |
| 12 | Variance | 9 |
| 13 | Energy (x, y, z, all) | 4 |
| | Total | 44 |

The extraction of maximum and minimum values across the X, Y, and Z dimensions signifies the peak magnetic interactions, correlating closely with the vehicle's size and type. The indices of these peaks provide insight into the timing of the vehicle's presence, which is crucial for understanding its speed and transit time. The length of the signal disruption offers an estimate of the vehicle's speed and dimensions, while the mean and

median values across all dimensions provide a thorough assessment of the vehicle's mass and overall magnetic profile. The counts of local maxima and minima, along with their mean values, reflect the vehicle's structural complexity, aiding in the differentiation of various vehicle types based on their magnetic signatures.

Variance features, denoted by 'V', differentiate between light, medium, and heavy vehicles. 'A' denotes the magnetic distortion along the X, Y, and Z axes, while 'S' indicates the sample size for each vehicle category. An average signal value is calculated for each vehicle type across the three axes, from which nine features per sample are extracted, as detailed in equations (2.26), (2.27), and (2.28), which describe the calculation of energy features ($Ex$, $Ey$, $Ez$, $Eall$) for the individual axes and in aggregate. 'L' represents the length of the vehicle's signal, and '$Xk$' denotes the signal values at time 'L'. Four energy features per sample are calculated by summing the squares of the signal values, normalized by the length.

$$\begin{cases} Average_{V_A} = \dfrac{\sum_1^S Mean(A)}{S} \\ Variance_{V_A} = \sqrt[2]{Mean(A) - Average_{V_A}} \end{cases} \tag{2.26}$$

$$\begin{cases} E_x = \dfrac{\sum_1^L X(k)^2}{L} \\ E_y = \dfrac{\sum_1^L Y(k)^2}{L} \\ E_z = \dfrac{\sum_1^L Z(k)^2}{L} \end{cases} \tag{2.27}$$

$$E_{all} = \frac{\sum_1^L (X(k)^2 + Y(k)^2 + Z(k)^2)}{L} \tag{2.28}$$

These features collectively create a multidimensional feature space crucial for ML algorithms to accurately classify vehicle types. They are carefully crafted to encompass the full magnetic signature of vehicles, thus improving the precision and dependability of the classification system within ITS. Such an approach markedly advances the vehicle identification process, which is essential for optimizing traffic flow and bolstering transportation security.

### 2.4.4 ANOVA F-test feature selection method

The model's performance depends on the quality of the inputs; higher-quality inputs are expected to yield better results. Feature selection methods play a crucial role in this context by eliminating or assigning lower scores to redundant features that lack relevance or predictive power for the target variable. The primary goals of these methods are to reduce model complexity, minimize noise, prevent overfitting, accelerate ML algorithms, and enhance performance outcomes. The Analysis of Variance (ANOVA) F-test method is utilized for this purpose [45], [46]. This statistical method compares the means of each feature across different classes of the target variable to determine if the differences are significant. The computed F-value for a feature, indicated in equation (2.29), is derived from the Mean Square Between (MSB), which is the quotient of the Sum of Squares Between Groups (SSB) and the degrees of freedom between groups, as specified in equation (2.30). The SSB itself is the aggregate of squared deviations of group means from the global mean, weighted by group size. The Mean Square Within (MSW) is obtained by dividing the Sum of Squares Within (SSW) by the degrees of freedom within groups, as specified in equation (2.31). The degrees of freedom, represented as $df$, are statistical values that quantify the amount of independent information pertinent to estimating another parameter. Specifically, $df_{between}$ between is $k-1$, where $k$ is the number of class groups, indicating the number of independent ways class means can vary. Similarly, $df_{within}$ is $N-k$, where $N$ is the total sample size, reflecting the number of independent pieces of information that can be utilized to estimate variability within these groups. $n_i$ represents the count of observations within $i$, $Xmean_i$ is the average of observations in group $i$, $Xmean$ is the overall mean of the data, and $x_{ij}$ is the $j$-th observation in group $i$.

$$F = \frac{MSB}{MSW} \tag{2.29}$$

$$MSB = \frac{SSB}{df_{between}} = \frac{\sum_{i=1}^{k} n_i (Xmean_i - Xmean)^2}{k-1} \tag{2.30}$$

$$MSW = \frac{SSW}{df_{within}} \frac{\sum_{i=1}^{k} \sum_{j=1}^{n_i} (X_{ij} - Xmean_i)^2}{N-k} \tag{2.31}$$

The F-value tests the null hypothesis, asserting that all group means are equivalent within the population. Higher F-values imply a greater degree of variability between groups relative to within them, signifying that the feature has strong discriminatory capabilities between classes. Among the 44 features extracted, they are ranked based on their importance using the ANOVA F-test method. In our experiments, the top N features (ranging from 1 to 44; N increases incrementally from 1 to 44) are selected. Each classification algorithm is then tested with these varying sets of selected features, as shown in Figure 2.10.

### 2.4.5 Focal cross-entropy loss function

The Focal loss (FL), as proposed by Lin et al. [47], [48], addresses the class imbalance problem by modifying the CE loss function. It introduces a modulating term that focuses training on challenging samples and less easy ones, regardless of their frequency. This strategy aims to shift the model's focus from abundant, simpler examples to scarcer, more complex ones. The main differences between CE and FL are the weighting factor (alpha, α) and the focusing parameter (gamma, γ). The gamma parameter intensifies the focus on difficult examples, while alpha adjusts for class imbalance by assigning a different weight to each class. The regular CE loss function is given by equation (2.32), where $p_t$ is the model's estimated probability for the class with the true label $t$, and $\alpha_t$ is a class-specific weighting factor. The FL introduces a modulating term to the CE loss, as shown in equation (2.33). The gamma parameter is tunable, decreasing the contribution from easy examples (where $p_t$ is high) and increasing it from difficult ones (where $p_t$ is low).

$$CE(p_t) = -a_t \log(p_t) \tag{2.32}$$

$$FL(p_t) = -a_t(1-p_t)^{\gamma} \log(p_t) \tag{2.33}$$

In FL, 'alpha' is pivotal for adjusting the weight given to each class, a critical aspect when dealing with imbalanced classes, unlike CE, which treats all classes uniformly. Meanwhile, gamma (γ) diminishes the loss contribution from easy examples—those for

which the model has high confidence—and emphasizes correction of the misclassified, difficult examples. In situations where an imbalanced dataset has one class significantly outnumbering the others, a model trained with CE might be biased towards the majority class. FL counteracts this by steering the training focus towards the minority class, which, though typically more challenging for the model to learn, is often crucial for accurate predictions. As illustrated in Figure 2.10 from the study [47], [48], when gamma (γ) is zero, FL is equivalent to CE, represented by the blue (top) curve. As gamma increases, the loss curve changes to further discount "easy" examples with lower loss values, thus, FL demonstrates substantial improvements over CE as gamma increases.

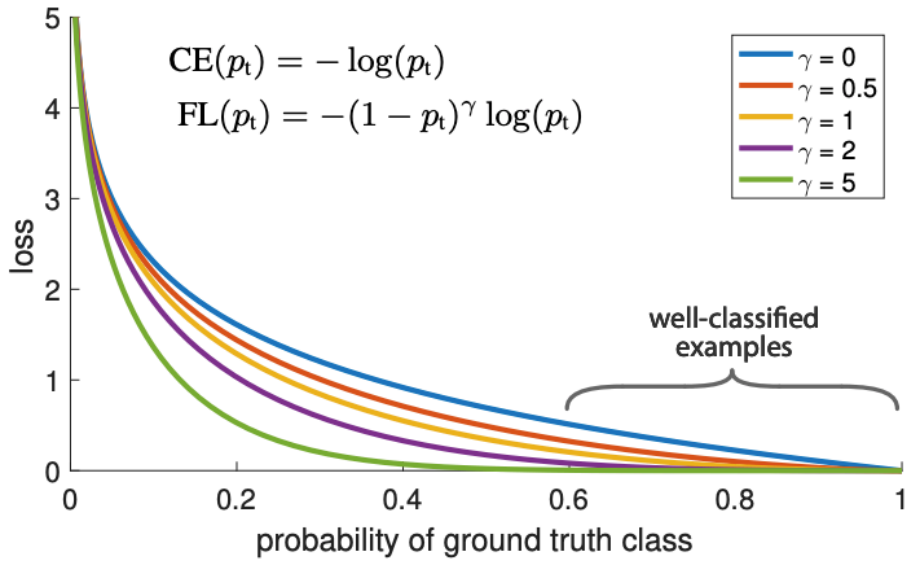


**Figure 2.10 Comparison of focal loss with cross-entropy loss across different values of focusing parameter $\lambda$ for imbalanced multi-class classification [47], [48].**

### 2.4.6 Grid search hyperparameter optimization

Hyperparameter optimization is a critical process that ensures the most appropriate parameters are selected for a classification model. Before the training phase begins, these parameters must be established; the model is then trained accordingly to these specifications. Models are trained using a training set and subsequently evaluated on a validation set for each hyperparameter configuration. The parameters yielding the best performance on the validation set are then used to retrain the model with the entire training set. Finally, this refined model is tested on the test set. For all classifiers, hyperparameter values are determined using the GS-CV, a method commonly adopted

for such tasks [49], [50]. Figure 2.11 illustrates the hyperparameter optimization process across two different hyperparameters for classifiers using grid search.

For the C4.5 classifier, fine-tuning entailed adjusting the 'min_samples_split', the minimum number of samples required to split an internal node; 'max_depth', the maximum depth that the tree can grow; and 'min_samples_leaf', the minimum number of samples that must be present at a leaf node. These adjustments aim to balance the model's sensitivity and mitigate the risk of overfitting. For the RF classifier, two parameters are varied: 'max_depth', the maximum depth that each tree in the forest can reach, and 'n_estimators', the total number of trees in the forest. This variation aims to assess their effects on model performance, focusing on enhancing accuracy without imposing excessive computational demands. Additionally, for XGBoost models, the learning rate—defined as the step size shrinkage used in updates—is adjusted from 0.1 to 1, along with 'max_depth', to regulate the training pace and reduce the risk of overfitting.

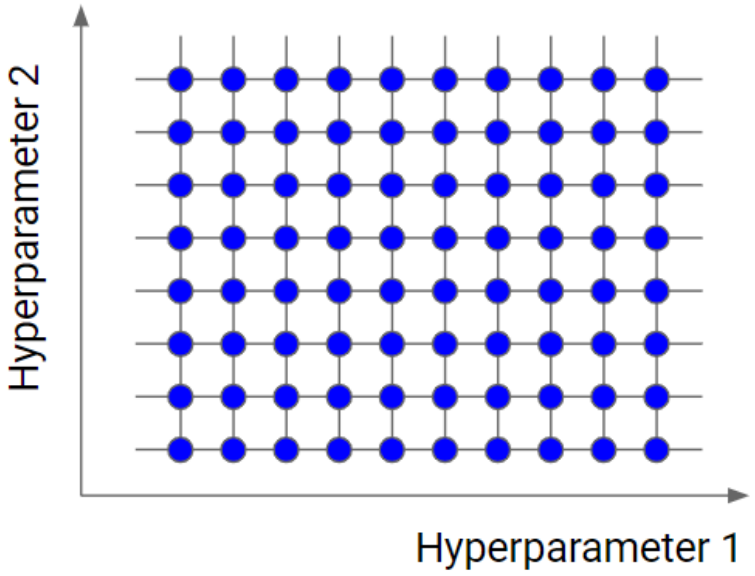


**Figure 2.11 Schematic representation of grid search for hyperparameter optimization across two different hyperparameters for classifiers [51].**

For the SVM model, two key parameters are varied: 'C', the regularization parameter that balances the trade-off between a smooth decision boundary and classifying training points correctly, and 'gamma', which defines the influence of a single training sample. These adjustments are made to assess their impact on model performance. Additionally, the selection of the most suitable 'kernel' function is crucial. The 'kernel'

function is responsible for transforming the input data into a higher-dimensional space, enabling a linear separation of the classes. This comprehensive hyperparameter tuning plays a vital role in enhancing the performance and accuracy of the SVM model. For the LR classifier, grid search was employed to optimize several parameters: 'penalty', specifying the norm used in penalization; 'C', representing the inverse of regularization strength; 'multi_class', defining the approach when dealing with more than two classes; and 'solver', indicating the algorithm used for optimization. These adjustments are crucial for ensuring efficient training convergence and effective handling of multiclass scenarios. The specific hyperparameters adopted for the classification algorithms in this study are detailed in Table 2.8, which outlines the optimization settings for each ML classifier involved in vehicle type classification.

**Table 2.8 Hyperparameter optimization settings for machine learning classifiers in vehicle type classification.**

| Classifier | Parameters |
|---|---|
| C4.5 | min samples split : [2–10]<br>max depth : [1–10]<br>min samples leaf : [1–5] |
| RF | max depth: 10, 20, 50, 80, 100<br>n estimator: 100, 200, 500, 1000 |
| LR | penalty: l1, l2, none<br>C: logspace(−4, 4, 20)<br>multi class: auto, ovr, multinomial<br>solver: newton-cg, lbfgs, liblinear, sag, saga |
| SVM | C : 0.1, 1, 10, 100, 1000<br>gamma : 1, 0.1, 0.01, 0.001, 0.0001<br>kernel : linear, rbf, poly, sigmoid |
| DNN | neurons : 32, 64, 128<br>neurons2 : 32, 64, 128<br>dropout rate: 0.1, 0.3, 0.5<br>dropout rate2: 0.1, 0.3, 0.5<br>learning rate: 10e−2, 10e−3, 10e−4<br>batch size: 2, 4, 6, 8 |
| XGBoost | learning rate = [0.1–1]<br>max depth : [1–20] |

### 2.4.7 Experiments

In this experiment, a camera and a 3-D magnetic sensor node are utilized to generate the dataset. The camera records videos, while the 3-D magnetic sensor node records magnetic disturbances caused by vehicles, and these processes occur concurrently. The labeling of magnetic disturbances is based on the recorded video. The first dataset, herein referred to as 'signal data', was obtained after applying zero-padding to the raw signal collected by the magnetic sensor node. From the raw signal, new features were extracted, resulting in a second dataset comprising a total of 44 features, which is herein referred to as 'extracted data'. Both datasets were normalized to a range between 0 and 1.

The classification process is applied to two datasets across three different variations: (i) using the signal data, (ii) using the extracted data, and (iii) using the extracted data augmented with the best-selected N features for each classifier. For the third variation, the extracted data is used, and the ANOVA F-test method is applied to rank the features according to their importance. The optimal Best-N features for each classifier are determined through an exhaustive search. Six classification algorithms, including C4.5, RF, LR, XGBoost, SVM, and DNN, are implemented in Python [52], utilizing the Scikit-Learn [53] and Keras [54] libraries. The datasets are divided into two parts: 70% for the training set and the remaining 30% for the test set. During model building, a 5-fold CV is applied to the training set. The hyperparameters for the models are determined using a portion of the training set, often referred to as the validation set. After identifying the best parameters, the model is retrained using these parameters on the entire training set. Subsequently, this finalized model is tested on the test set. Figure 2.12 presents a flowchart of the classification process, and Figure 2.13 illustrates the methodology for identifying the best N features for each classifier.

The default DNN architecture consists of three layers with 32, 32, and 3 neurons, respectively. The first two layers utilize the ReLU activation function, while the final layer uses a SoftMax activation function. Kernel initializers are set to Glorot Uniform with a specific seed to ensure reproducible outcomes. To prevent overfitting, batch normalization and dropout rates are implemented after each layer. The Adam optimizer is employed in conjunction with the FL function to mitigate class imbalance. The batch size is set at 8, and the number of epochs is limited to 30, with an early-stopping mechanism in place to halt training if no validation accuracy improvement is observed

after five epochs and to restore the best model weights. The details of the DNN's layer configuration are provided in Table 2.9.

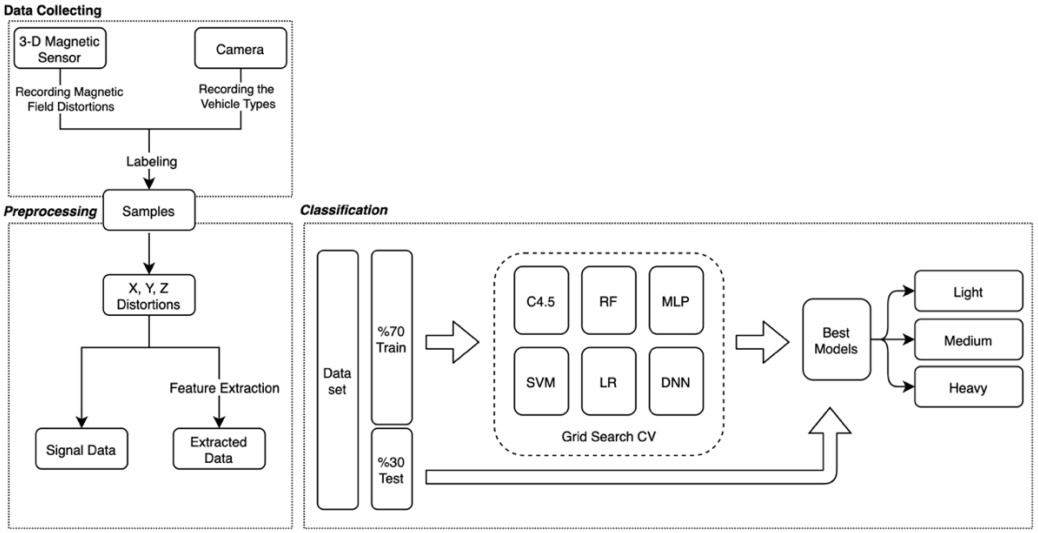


**Figure 2.12 Flowchart of the vehicle type classification process.**

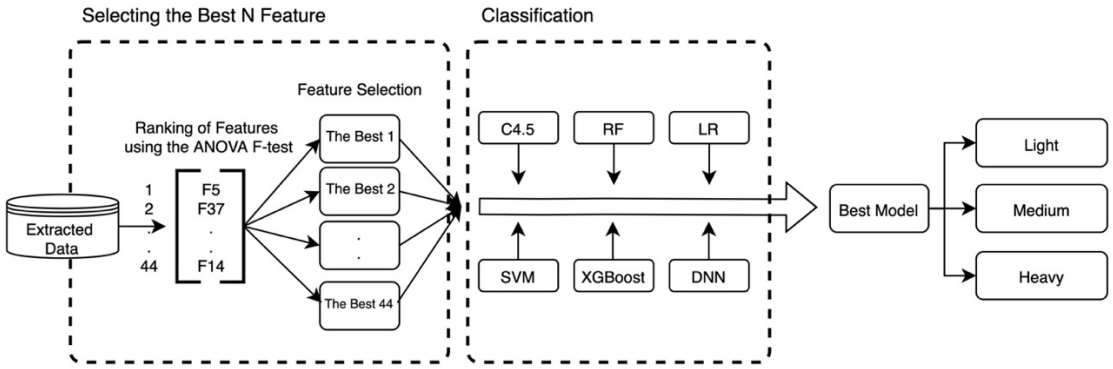


**Figure 2.13 Flowchart of the selection of the best N features for each classifier.**

**Table 2.9 Configuration of Deep Neural Network layers for vehicle type classification.**

| Layer names | DNN models |
|---|---|
| L1 | Dense (32) |
| L2 | Batch normalization |
| L3 | Dropout (0.1) |
| L4 | Dense 1 (32) |
| L5 | Batch normalization 1 |
| L6 | Dropout 2 (0.1) |
| L7 | Dense 2 (3) |

## 2.5 Results and Discussion

### 2.5.1 Classification methods

This chapter presents a comprehensive performance evaluation of various classification methods, including ML and DL techniques, applied to a dataset comprising 376 samples. The focus of this analysis is on key performance metrics: accuracy, precision, recall, and F1-measure. This evaluation is particularly crucial due to the imbalanced class distribution within the vehicle dataset. Tables 2.10, 2.11, and 2.12 display a comparison of classifiers under both default parameter settings and hyperparameter optimization across three distinct scenarios.

**Table 2.10 Performance results of the machine learning classifier under default and grid search cross-validation hyperparameter settings on signal dataset.**

| Hyperparameter | # of features | Classifier | ACC (%) | PRE (%) | REC (%) | F1 (%) |
|---|---|---|---|---|---|---|
| Default | 621 | C4.5 | 78.76 | 81.58 | 71.15 | 79.76 |
| | | RF | 82.30 | 83.34 | 67.68 | 82.75 |
| | | LR | 78.76 | 79.42 | 66.08 | 78.90 |
| | | SVM | 83.18 | 83.13 | 62.03 | 82.17 |
| | | XGBoost | **84.95** | 84.84 | 70.81 | **84.88** |
| | | DNN | 78.76 | 79.42 | 66.08 | 78.90 |
| GS-CV | 621 | C4.5 | 78.76 | 81.58 | 71.15 | 79.76 |
| | | RF | 82.30 | 83.34 | 67.68 | 82.75 |
| | | LR | 83.18 | 82.82 | 64.99 | 82.39 |
| | | SVM | 83.18 | 82.82 | 64.99 | 82.39 |
| | | XGBoost | 84.84 | **85.05** | **72.71** | 84.84 |
| | | DNN | 79.64 | 79.94 | 64.55 | 79.78 |

The study commences with an examination of classifiers using a signal dataset under both default parameter settings and GS-CV hyperparameter optimization. The performance results of the classifier are shown in Table 2.10. Notably, under the default parameter, the XGBoost classifier consistently demonstrates superior performance, achieving remarkable accuracy of 84.95% and F1-measure scores of 84.88%. This performance indicates XGBoost's robustness in handling the dataset without any specific parameter tuning. While under GS-CV optimization, there is a slight decrease in accuracy

and F1-measure, this is counterbalanced by an increase in precision and recall metrics. The reduction in accuracy and F1-measure suggests that the model, with optimized hyperparameters, becomes slightly more conservative in its predictions. This could be a result of the model being tuned to avoid overfitting, leading to a more generalized approach to handling the dataset. Such a scenario is often witnessed when the model is adjusted to be less sensitive to the noise within the training data, thereby potentially missing out on some true positives, which affects both accuracy and the F1-measure.

**Table 2.11 Performance results of the machine learning classifier under default and grid search cross-validation hyperparameter settings on extracted dataset.**

| Hyperparameter | # of features | Classifier | ACC (%) | PRE (%) | REC (%) | F1 (%) |
|---|---|---|---|---|---|---|
| Default | 44 | C4.5 | 82.30 | 82.30 | 67.78 | 82.19 |
| | | RF | 82.30 | 82.70 | 65.67 | 82.49 |
| | | LR | 76.99 | 66.32 | 41.46 | 71.15 |
| | | SVM | 78.76 | 68.18 | 45.16 | 72.95 |
| | | XGBoost | 84.95 | 85.47 | 65.84 | 84.68 |
| | | DNN | 77.87 | 66.71 | 35.91 | 70.34 |
| GS-CV | 44 | C4.5 | 84.95 | 84.45 | 69.86 | 84.53 |
| | | RF | 82.30 | 82.70 | 65.67 | 82.49 |
| | | LR | **89.38** | 89.38 | 77.75 | 89.38 |
| | | SVM | **89.38** | **89.82** | **80.71** | **89.56** |
| | | XGBoost | 87.61 | 87.76 | 72.99 | 87.61 |
| | | DNN | 82.30 | 82.30 | 66.73 | 82.30 |

Table 2.11 compares the performance of classifiers under both default and GS-CV settings on an extracted dataset. Under default parameters, the XGBoost classifier notably stands out, achieving an accuracy of 84.95% and an F1-measure of 84.68%, which suggests its effectiveness in utilizing the extracted features. When subjected to GS-CV optimization, both the SVM and LR classifiers exhibit remarkable improvements. Specifically, the SVM reaches a high accuracy of 89.38% and an F1-measure of 89.56%, while the LR demonstrates significant increases in accuracy and precision. This marked improvement under GS-CV optimization indicates that the SVM and LR classifiers benefit substantially from hyperparameter tuning, particularly with the extracted dataset. This outcome suggests that optimal hyperparameter tuning can unlock the full potential

of these models, especially when aligned with the comprehensive feature set extracted from the dataset.

In general, as observed in Table 2.11, this strategic feature extraction significantly enhances the performance of nearly all the techniques. The extracted features, carefully derived from the 3-D vehicle signal, offer a rich and nuanced perspective of the dataset, capturing essential characteristics of the vehicles. When these detailed features are coupled with GS-CV, an optimization method that meticulously tunes hyperparameters, the classifiers demonstrate an augmented ability to interpret and analyze the data effectively.

**Table 2.12 Performance results of the machine learning classifier under default and grid search cross-validation hyperparameter settings on extracted dataset with different number of features.**

| Hyperparameter | # of features | Classifier | ACC (%) | PRE (%) | REC (%) | F1 (%) |
|---|---|---|---|---|---|---|
| Default | 25 | C4.5 | 84.07 | 85.41 | 74.45 | 84.52 |
| | 11 | RF | 84.95 | 87.25 | 78.73 | 85.75 |
| | 5 | LR | 78.76 | 68.18 | 45.16 | 72.95 |
| | 18 | SVM | 84.95 | 85.73 | 64.79 | 84.20 |
| | 15 | XGBoost | 86.72 | 87.19 | 69.55 | 86.42 |
| | 17 | DNN | 86.72 | 86.09 | 67.65 | 86.15 |
| GS-CV | 25 | C4.5 | 84.95 | 84.45 | 69.86 | 84.53 |
| | 10 | RF | 85.84 | 88.63 | 83.12 | 86.60 |
| | 27 | LR | 89.38 | 89.38 | 77.75 | 89.38 |
| | 20 | SVM | 90.26 | 91.08 | 84.04 | 90.59 |
| | 30 | XGBoost | 88.49 | 88.72 | 74.31 | 88.43 |
| | 30 | DNN | **91.15** | **91.95** | **84.41** | **91.50** |

In a third scenario, the ANOVA F-test feature selection method is applied to determine the most significant features for vehicle classification tasks on the extracted dataset. The resulting feature rankings are used to enhance the performance of ML and DNN algorithms. From Table 2.12, it is evident that the performance of classifiers such as C4.5, LR, SVM, XGBoost, and DNN has been enhanced when operating optimized hyperparameter settings obtained through GS-CV. The DNN and SVM classifiers, in particular, have shown exceptional improvement post-optimization, indicating that these

algorithms benefit significantly from feature selection and hyperparameter tuning. For instance, under default settings with 15 selected features, XGBoost achieves an accuracy of 86.72% and an F1-measure of 86.42%. However, when optimized through GS-CV with 30 features, the DNN classifier achieves a superior accuracy of 91.15% and an F1-measure of 91.50%. The most effective 30 features are detailed in Table 2.13. In conclusion, this study has demonstrated that feature selection through an ANOVA F-test and careful hyperparameter tuning through GS-CV can significantly elevate the efficacy of ML and DNN classifiers in a vehicle classification task.

Observations indicate that certain extracted features, including the mean of the signal, variance, and energy, significantly impact performance, particularly on specific axes. For variance features, only the Z-axis values are utilized. Additionally, attributes such as the maximum and minimum values and their indices, the length of the signal, and the count of local maxima and minima are pivotal across all axes. The performance results suggest that classifiers produce improved outcomes when combined with feature extraction, the best N feature selection, and hyperparameter optimization. The specific hyperparameters optimized for each classifier are delineated in Table 2.14.

**Table 2.13 The best 30 features selected for the Deep Neural Network classifier.**

| No | Features | # |
|---|---|---|
| 1 | Maximum values (x, y, z) | 3 |
| 2 | Index of maximum (x, y, z) | 3 |
| 3 | Minimum values (x, y, z) | 3 |
| 4 | Index of minimum (x, y, z) | 3 |
| 5 | Length of signal (l) | 1 |
| 6 | Mean of the signals (x) | 1 |
| 7 | Median of the signals (x, z) | 2 |
| 8 | # of local maximum (x, y, z) | 3 |
| 9 | # of local minimum (x, y, z) | 3 |
| 10 | Mean of local maximum (x, z) | 2 |
| 11 | Mean of local minimum (x, y, z) | 3 |
| 12 | Variance | 1 |
| 13 | Energy (x, all) | 2 |
| | Total | 30 |

This comprehensive evaluation of ML and DL classifiers in vehicle type classification reveals that hyperparameter optimization, coupled with strategic feature

extraction and selection, significantly enhances model performance. The XGBoost classifier emerges as a standout performer, demonstrating consistent strength across various scenarios. The notable improvements seen in SVM and LR under GS-CV optimization highlight the impact of fine-tuning model parameters. Furthermore, the DNN's marked performance improvement with the top 30 features underscores the importance of targeted feature selection in dealing with complex, high-dimensional datasets. Overall, this study contributes valuable insights into the effective application of ML algorithms in vehicle type classification, emphasizing the critical role of methodical feature selection and hyperparameter tuning in optimizing classifier performance. The findings have important implications for future research in ML, particularly in scenarios involving imbalanced datasets and the need for precise, nuanced classification.

**Table 2.14 Optimum hyperparameters configurations for each machine learning classifier using grid search cross-validation technique.**

| Classifier | Parameters |
|---|---|
| C4.5 | min samples split : 4, max depth : 4, min samples leaf : 1 |
| RF | max depth: 20, n estimator: 100 |
| LR | penalty: l1, C: 4.281, multi class: auto, solver: saga |
| SVM | C: 100, Gamma: 0.1, kernel : rbf |
| DNN | Neurons: 128, neurons2 : 32, dropout rate: 0.1, dropout rate2: 0.3 learning rate: 10e−2, batch size: 8 |
| XGBoost | learning rate = 0.3, max depth: 7 |

### 2.5.2 Battery lifetime

In this research, magnetic changes over the 3-D magnetic sensor nodes are detected using a vector magnitude-dependent measurement method as vehicles pass. To sample vehicle movement, a time-dependent measurement method is employed. The sensor's current consumption in both sleep and active states (activated when a vehicle passes) is measured using a power analyzer named EnergyTrace. It is observed that current consumption during the sleep state is significantly low. However, communication between the sensor and the gateway notably increases battery consumption. To address this, a data aggregation technique is utilized in the communication process. This technique optimizes power consumption by transmitting maximum information in minimally sized packets. Additionally, Figure 2.14 illustrates the relationship between

the sensor node's lifetime and the number of samples collected from one hundred vehicles per day. With an average of 53 samples obtained per vehicle in the dataset, the current consumption characteristics of the magnetic sensor node suggest that the proposed node can operate for up to two years without the need for new batteries.

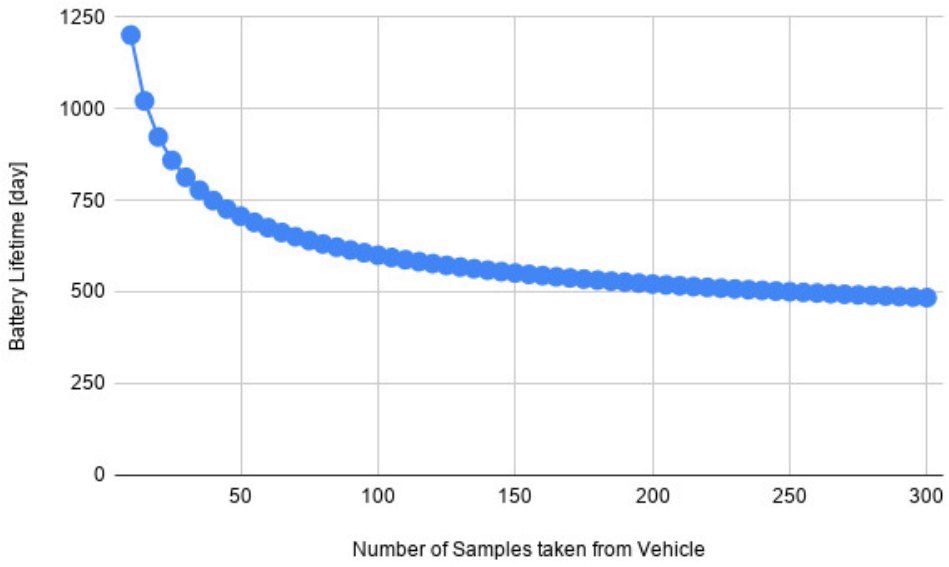


**Figure 2.14 Battery lifetime based on the number of samples taken from the vehicle.**

# Chapter 3

# Deep learning approaches for vehicle type classification with 3-D magnetic sensor

## 3.1 Motivation

In recent years, rapid population growth and vehicle demand have significantly increased the number of vehicles in traffic, particularly in metropolitan cities. This situation is increasing the need for ITS. Vehicle detection and classification have become an indispensable part of ITS to increase human comfort, improve traffic management, and enable future development of transport infrastructure. Significant investments are being made and used in the development, implementation, and maintenance of traffic monitoring systems in many countries [10]. Recently, in spite of the technical challenges, several vehicle type classification systems have been developed based on accelerometers [11], acoustic sensors [14], cameras [31], hybrid methods [32], loop detectors [15], LIDAR [17], piezoelectric sensors [18], vibration sensors [19], and magnetic sensors [21-30]. These technologies have important specific characteristics and requirements, such as sensor types, hardware settings, setup processes, parameter settings, operating environments, weather and noise resistance, battery lifetime, and even maintenance and installation costs.

In the literature, it is shown that magnetic sensors are preferred for vehicle classification due to their advantages, such as strong climate adaptation, small size, easy installation, and low cost. Specifically, the study [26] focused on the classification of similar vehicle sizes using multiple sensor nodes and utilized an XGBoost method. The study [27] proposed the KNN method for the classification of vehicle types. In [28], the magnetic waveforms obtained from two sensor nodes were fussed over, and a SVM was used for classification. The study [30] focused on classifying vehicles with the CNN

method. In contrast to the existing studies and the previous work, to the best of our knowledge, this chapter is the first study focusing on DL methods and soft voting ensemble techniques to classify vehicle types with a single 3-D magnetic sensor node. The performance comparison reveals that the suggested soft voting ensemble technique, which ensembles DL classifiers, enhances both accuracy and f-measure scores, achieving improvements of 92.92% and 93.42%, respectively.

The rest of the chapter is organized as follows: Section 3.2 elaborately describes the DL methodologies and the proposed approach. Then, Section 3.3 shows the performance results of the models comparatively and discusses the outcomes.

# 3.2 Methods

## 3.2.1 Synthetic minority oversampling technique (SMOTE)

The main reason for applying the oversampling method is that the number of samples is insufficient to train DL models, and the unbalanced class problem causes poor performance. In this study, the Synthetic Minority Oversampling Technique (SMOTE) is applied [55] to increase the number of samples in the training set. The SMOTE algorithm randomly generates new minority samples using the following rules: First, $X_i$ sample is randomly selected from the minority classes; then the five samples are determined based on the KNN of the selected $X_i$ data; and finally, randomly $X_k$ sample is selected. The new synthetic data $X_{new}$ is between the $X_i$ and $X_k$ samples according to the $\lambda$ values, which is randomly selected between (0,1). The formula is shown in equation (3.1) below:

$$X_{new} = X_i + \lambda(X_k - X_i) \tag{3.1}$$

The sample sizes of the original vehicle dataset and the new sample sizes of the vehicle types are shown in Tables 3.1 and 3.2. In the training dataset, minority classes (light and heavy) are processed by the SMOTE algorithm and generate new synthetic samples, as shown in Figures 3.1 and 3.2 for light and heavy vehicles, respectively. During the analysis of the new synthetic samples, no absurdity was observed in the waveforms.

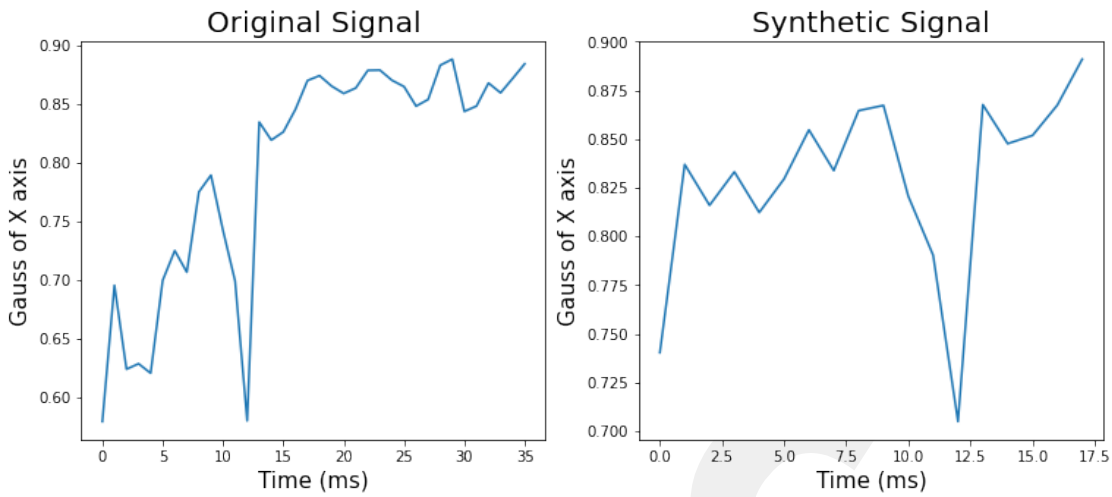


**Figure 3.1 The X-axis representations of original and synthetic signals (processed by the SMOTE Algorithm) for light vehicles.**

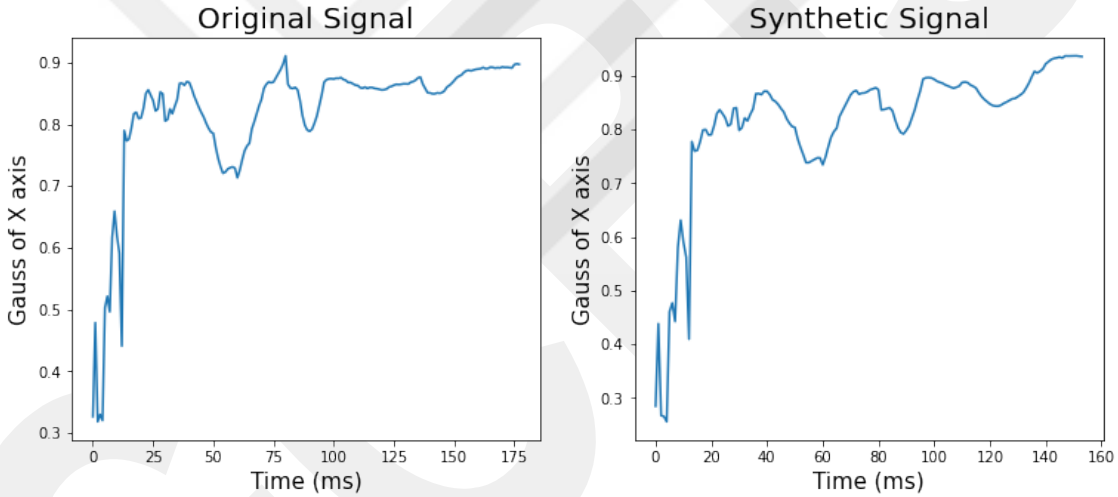


**Figure 3.2 The X-axis representations of original and synthetic signals (processed by the SMOTE Algorithm) for heavy vehicles.**

**Table 3.1 Distribution of vehicle types in training and test sets prior to oversampling technique.**

| Vehicle Type | Light | Medium | Heavy |
|---|---|---|---|
| Train | 33 | 207 | 23 |
| Test | 14 | 89 | 10 |
| Total | 47 | 296 | 33 |

**Table 3.2 Distribution of vehicle types in training and test sets after oversampling technique.**

| Vehicle Type | Light | Medium | Heavy |
|---|---|---|---|
| Train | 207 | 207 | 207 |
| Test | 14 | 89 | 10 |
| Total | 221 | 296 | 217 |

### 3.2.2 Two-dimensional multi-color visualization

Transfer learning is a powerful network that tends to learn edges, textures, patterns, and object parts in images [56] and yields satisfactory results. Therefore, in this study, vehicle signals are converted into 2-D images in order to perform vehicle classification by utilizing transfer learning models. Firstly, the lengths of samples are scaled to the maximum signal length of 207. The next step is to color the area under the X, Y, and Z curves. Each color represents a different condition. For example, the pink color represents the area under the X curve and the upper area of the Y and Z curves. Table 3.3 shows the combination of conditions during the coloring process. The converted images become 216 × 216 colored images, and the conversion of the signals to the images for the light, medium, and heavy vehicles are shown in Figures 3.3, 3.4, and 3.5. Moreover, the study explored the conversion of signals into visual representations employing lines and bars; however, this approach was selected over others due to its superior performance.

**Table 3.3 Legend of Colors for Representation of Conditions Along X, Y, and Z Axes.**

| No | Condition (Under Axis-Curve) | Color |
|---|---|---|
| 1 | - | Blue |
| 2 | X | Purple |
| 3 | Y | Red |
| 4 | Z | Green |
| 5 | X, Y | Black |
| 6 | X, Z | Olive |
| 7 | Y, Z | Turquoise |
| 8 | X, Y, Z | White |

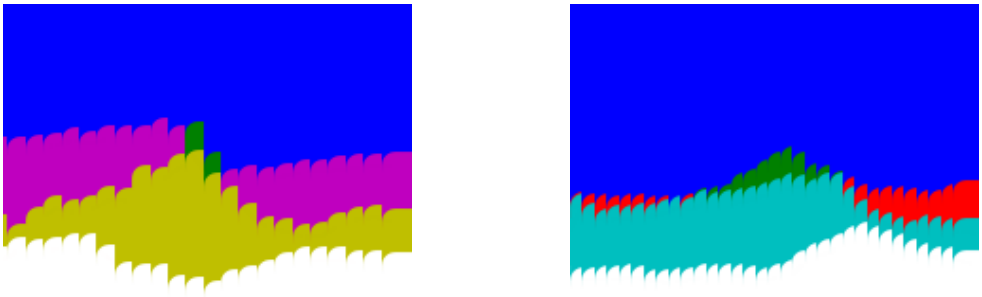

**Figure 3.3 Examples of multi-color visualization of axis-curve data for light vehicles.**

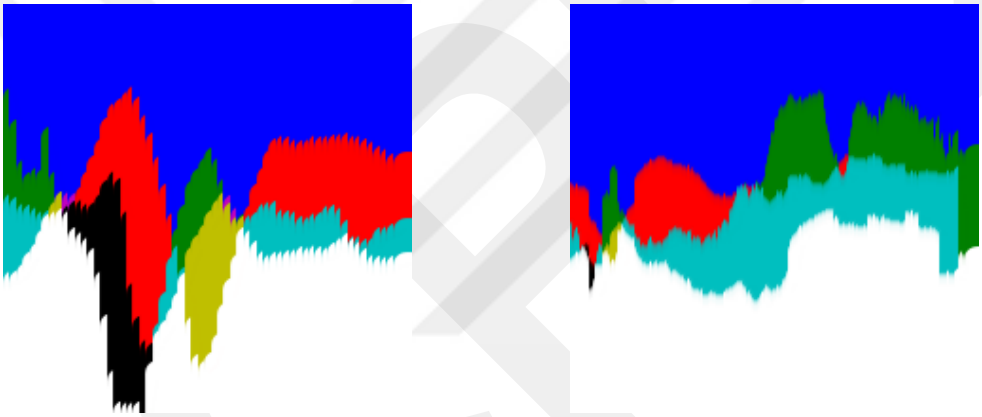

**Figure 3.4 Examples of multi-color visualization of axis-curve data for medium vehicles.**

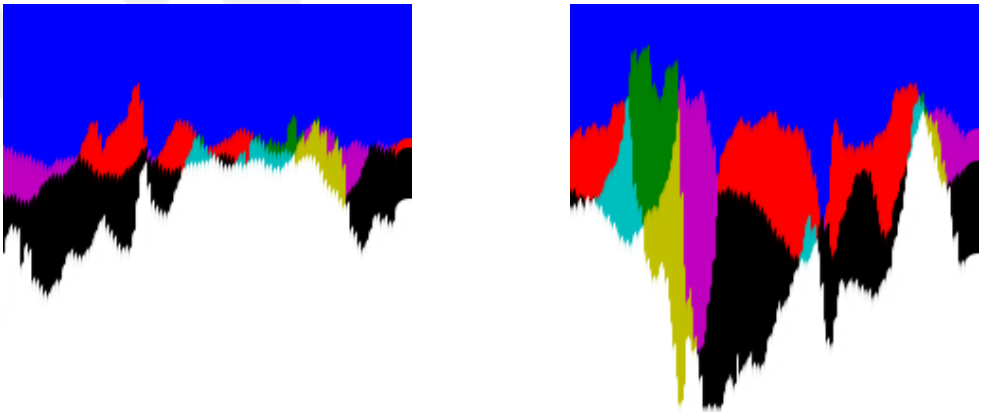

**Figure 3.5 Examples of multi-color visualization of axis-curve data for heavy vehicles.**

Importantly, the shape of the dataset, i.e., ‘‘samples, features’’, is not suitable for the LSTM and GRU methods. The shape of the dataset is reshaped into ‘‘samples, time-steps, and features’’ which means one sequence is one sample, one sequence has multiple time steps according to the length of the vehicle signal, and one feature corresponds to one variety of signal. The reshaped dataset is ‘‘sample size, 207, 3’’, in which 207 means the vehicle signal length and 3 means the X, Y, and *Z*-axis.

### 3.2.3 Support vector machine (SVM)

It is a simple and effective ML method often used by the community for classification, regression, and outlier detection. It has a simple method for separating classes, which draws parallel lines between classes. Also, hyper parameter optimization is applied, which ensures the classifiers are trained with the most suitable parameters. In the implementation of the SVM classifier, hyperparameter optimization is applied using the grid search method. The parameters are determined before the classification method, and the best parameters are obtained using the training and validation sets, and the optimal parameters are used on the test set. The hyperparameters are optimized for the SVM classifiers as follows: C = {0.1, 1, 10, 100, 1000}, gamma = {1, 0.1, 0.001, 0.0001}, and kernel = {linear, rbf, poly, sigmoid}.

### 3.2.4 Recurrent neural networks (RNN)

In traditional neural networks, each observation is considered independent, as the networks do not retain past or historical information; they lack memory of previous events. A Recurrent Neural Network (RNN) introduces an internal loop, enabling it to process sequential data effectively. This capability is crucial for tasks such as language modeling, text generation, speech recognition, and time-series analysis, where current observations are influenced by previous ones, making them interdependent. As shown in Figure 3.6, an RNN processes a sequence of inputs, such as a vehicle signal, across different timesteps, considering both current and past data points for comprehensive understanding. Unlike feedforward networks, an RNN's hidden layer output at any given timestep is informed by the data from preceding timesteps. The output layer then generates the final output, which can vary from a single value at the sequence's end for classification tasks to a series of values for tasks like sequence labeling.

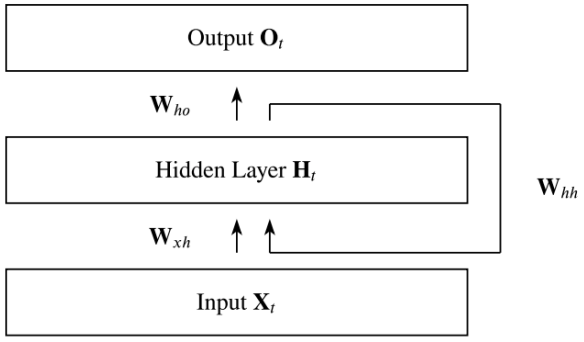


**Figure 3.6 Basic illustration of the recurrent neural networks [57].**

Figure 3.4 illustrates basic RNN architecture. At each timestep $t$, the standard RNN cell takes two inputs: the current input $X_t$, and the hidden state from the previous timestep $h_{t-1}$. The updated hidden state $h_t$ is computed using the equation (3.2), incorporating the previous timestep's hidden state weight matrix $W_{hh}$, the current input weight matrix $W_{xh}$, a bias term $b_h$, and the hyperbolic tangent function $tanh$, which adds non-linearity and outputs values between -1 and 1. This choice of activation function is not exclusive, and other non-linear functions can be employed as required by the model design. The output $o_t$ at timestep $t$ is typically calculated using the hidden state $h_t$ as shown in equation (3.3), with $W_{ho}$ being the hidden-to-output weight matrix and $b_o$ the output bias.

$$h_t = \tanh\left(W_{hh} \cdot h_{t-1} + W_{xh} \cdot x_t + b_h\right) \tag{3.2}$$

$$o_t = W_{ho} \cdot h_t + b_o \tag{3.3}$$

RNNs face the challenge of the vanishing gradient problem, where gradients can shrink to the point that learning becomes extremely slow or halts. This issue hinders the network's ability to capture long-range dependencies within the data. To overcome this, advanced RNN variants like LSTM networks and GRUs have been developed. Figure 3.7 shows an illustration of the architectural differences between RNN, LSTM, and GRU.

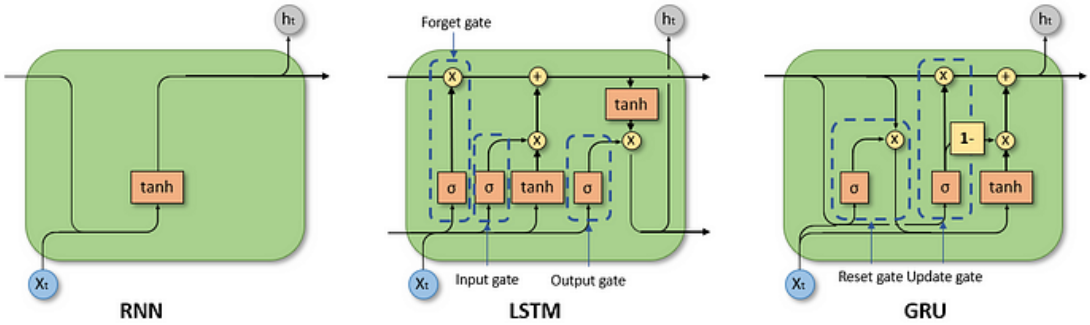


**Figure 3.7 Illustration of the architectural differences between recurrent neural networks, a long short-term memory, and a gated recurrent unit [58].**

### 3.2.5 Long short-term memory (LSTM)

Long Short-Term Memory networks [59], [60], commonly known as LSTMs, are an advance type of RNN algorithm designed to address the vanishing gradient descent problem and are adept at capturing long-range dependencies in data sequences. The unique architecture of LSTMs incorporates a set of gates that modulate the memory flow across time steps. The forget gate $f_t$ determines which information to discard from the cell state as shown in equation (3.4), the input gate $i_t$ selects new data to update the cell state as shown in equation (3.5), and the output gate $o_t$ controls the output based on the cell state for the current time step as captured in equation (3.6). These gates are updated at every time step $t$ by integrating the current input $x_t$, the preceding hidden state $h_{t-1}$, the former cell state $C_{t-1}$, and employing weight matrices ($W$ and $U$), and bias vectors $b$. The sigmoid function ($\sigma$) plays a crucial role in this process by constraining the output values between 0 and 1. The candidate cell state $\tilde{C}_t$ combines the present input with the prior hidden state, as indicated in equation (3.7). The new cell state $C_t$ evolves by selectively forgetting and acquiring new information, formalized in equation (3.8). The updated hidden state $h_t$ is then computed as shown in equation (3.9), utilizing the tanh function to scale the outputs.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{3.4}$$

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \tag{3.5}$$

$$o_t = \sigma\,(W_i \cdot [h_{t-1}, x_t] + b_o) \tag{3.6}$$

$$\tilde{C}_t = tanh\,(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{3.7}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C} \tag{3.8}$$

$$h_t = o_t * \tanh(C_t) \tag{3.9}$$

The LSTM's ability to maintain a cell state through time and its use of gates to control the flow of information make it ideal for tasks where understanding context and remembering things from the past are essential, such as in language modeling and time series prediction.

### 3.2.6 Gated recurrent unit (GRU)

Gated Recurrent Units (GRUs) [61] are a streamlined variant of RNN that are structured to efficiently capture dependencies across varying time intervals in sequence data. Similar to LSTM units but with fewer parameters, GRUs manage the flow of information without the need for separate memory cells through a duo of gating units. Update Gates controls the extent to which information from the previous state is carried over to the current state, as shown in equation (3.4). It acts as a regulator for updating the unit's activations and is analogous to the combined functionality of the forget and input gates in an LSTM. The reset gate determines the amount of prior information to be disregarded. It essentially modulates the influence of the previously computed state on the current state's candidate activation, as shown in equation (3.5). The update gate $z_t$ works to control the degree to which a GRU unit updates its content, dictating how much of the past information should persist. The reset gate $r_t$ allows the GRU to forget the previously computed state, influencing how much of the past state is relevant for the current state calculation. The candidate hidden state is calculated by $\tilde{h}_t$, which is a combination of the current input and the previously computed state, modulated by the reset gate as shown in equation (3.7). The final updated state is $h_t$, blending the old state with the new candidate state, based on the update gate's output. In these equations, $x_t$ represents input at time $t$, $h_{t-1}$ is the hidden state from the previous time step, $W$, $W_z$, $W_r$,

$U$, $U_z$, $U_r$ are weight matrices, $b_z$ , $b_r$ , $b_h$ are bias terms, $*$ denotes element-wise multiplication.

$$z_t = \sigma\,(W_z \cdot x_t + U_z \cdot h_{t-1} + b_z) \tag{3.4}$$

$$r_t = \sigma\,(W_r \cdot x_t + U_r \cdot h_{t-1} + b_r) \tag{3.5}$$

$$\tilde{h}_t = tanh\,(W \cdot x_t + U \cdot (r_t * h_{t-1}) + b_h) \tag{3.7}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \tag{3.8}$$

This gating mechanism helps GRUs to capture long-range dependencies and avoid the vanishing gradient problem common in standard RNNs. GRUs have been shown to perform on par with LSTMs on certain tasks, with a simpler model that can be easier to modify and run faster due to fewer parameters.

### 3.2.7 Transfer learning

Transfer learning is a technique that leverages knowledge acquired from prior tasks to inform a new task [62]. This technique is notably effective in transferring learned features from one domain to another, especially valuable in scenarios with limited data availability [56]. Transfer learning enhances learning efficacy and task performance by employing features, weights, and biases developed through training on extensive datasets. The process involves applying the insights obtained from solving a primary problem (task 1) to a secondary, related problem (task 2). Initially, a model is trained on task 1, forming the foundational knowledge for task 2. Subsequently, for task 2, a new model is constructed, adopting both the architecture and the learned parameters from the successful segments of the original model. This new model is then fine-tuned using the specific dataset for task 2. This methodical application of transfer learning is elucidated in Figure 3.8. The choice of transfer learning model depends on the specific requirements of the task. While these models may be pre-trained on the same dataset, like ImageNet, they offer a range of architectural choices and trade-offs.

In the realm of transfer learning for DNNs, three primary strategies could be employed: First, the convolutional base, pretrained on extensive datasets like ImageNet,

is frozen, and only the upper layers, typically fully connected layers, are retrained to adapt to the new task. Secondly, a more nuanced approach involves partially unfreezing the convolutional base, particularly the last layers, and jointly retraining them with the top layers, thereby fine-tuning the model to align more closely with the specific characteristics of the new data. Lastly, the model can be entirely retrained from the ground up, which, while more computationally intensive, allows for complete customization to the new task, leveraging no prior learned patterns or features [63]. Each method presents a unique balance between computational efficiency and the model's ability to adapt to the specific characteristics of the new task. The choice of method is often dictated by the size and nature of the new dataset, computational constraints, and the desired level of task specificity.

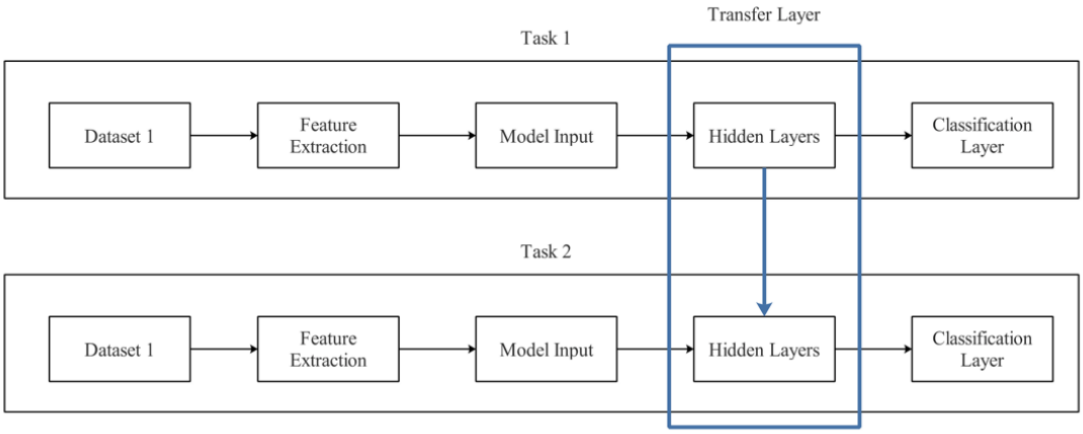


**Figure 3.8 Schematic diagram of a transfer learning model: The first phase involves training with Task 1, followed by training a new model for Task 2 that leverages the knowledge acquired from the model developed for Task 1 [63].**

## 3.3 Experiments

In this chapter, the test set is obtained by selecting 30% of the total dataset using stratified random sampling, and the remaining samples are used as a training set. The SMOTE method is applied to increase the number of samples in the training set while keeping the size of the test set same. In addition, for the hyperparameter optimization, 30% of the new training set is randomly selected for the validation set, and the rest is stored as a training set for the SVM classifier. Others DL methods have been applied directly. For the LSTM and GRU models, we reshaped the dataset, and for the transfer

learning model, we converted the vehicle signals to 2-D images. For this purpose, VGG16, VGG19, Xception, MobilNet, MobilNetV2, DenseNet121, DenseNet169, and DenseNet201 are implemented as DL models. The block diagram in Figure 3.9 outlines how the classification process in this study is conducted. Models are trained on the training set and validated on the validation set for DL models. Lastly, the model is evaluated for the test set. Individual results are obtained for each classifier. All ML approaches are implemented in the Python programming language. LSTM, GRU, and transfer learning approaches are implemented using Keras libraries [54], and the SVM method is implemented using scikit-learn libraries [53].

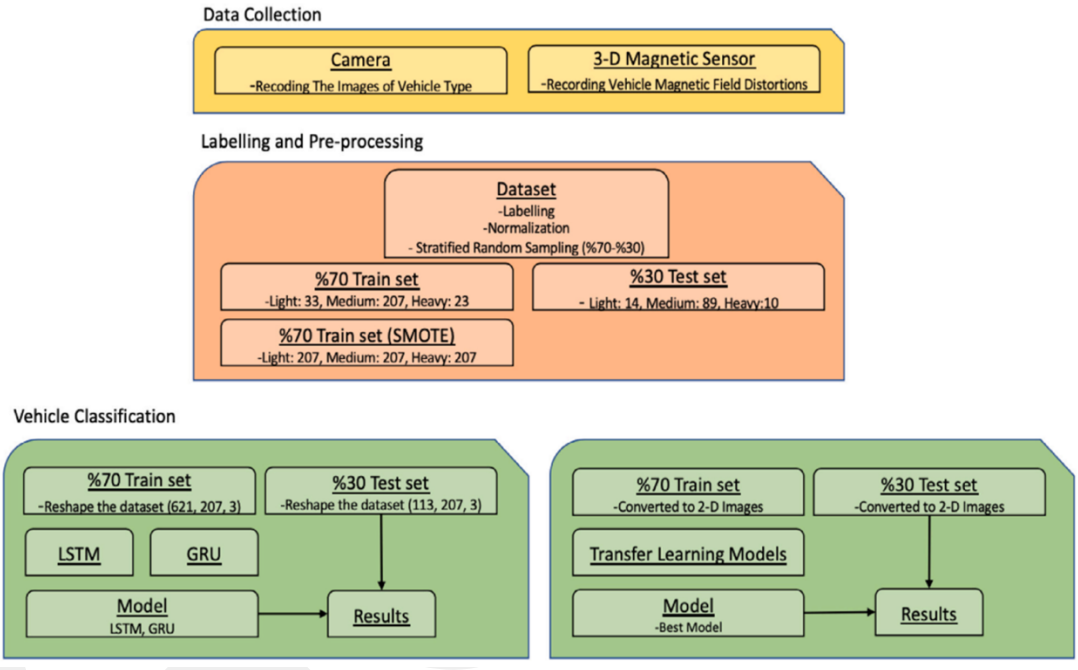


**Figure 3.9 Block diagram of the classification process.**

In traditional neural networks, each observation is treated independently, lacking the capability to retain or utilize historical data. This aspect becomes especially critical when dealing with data from 3-D magnetic sensors in vehicle signal analysis. In such scenarios, not only does each vehicle's signal vary in duration, but each momentary signal value is also intrinsically linked to its preceding value, creating a sequential dependency. This necessitates the use of models like RNNs, which can process such sequential data effectively. Figure 3.10 highlights the adaptability of RNNs and related sequence models for managing diverse data types and tasks. This includes the one-to-one model, typical of

standard neural networks; the one-to-many model, which produces a sequence of outputs from a single input; and the many-to-one model, where a series of inputs culminate in a single output. For vehicle type classification in this study, a many-to-one model approach is employed. The detailed architecture of the LSTM and GRU models used in this research is outlined in Table 3.4. Considering the multi-class nature of the classification problem, the SoftMax function is employed as the activation function for the final layer. Optimization is performed using the Adam optimizer with a learning rate of 0.0001 and the categorical CE loss function. The mini-batch size is set at 16, with an epoch limit of 200. ModelCheckpoint is utilized for saving model weights during training, with the optimal epoch for the test set determined by validation loss. Batch normalization and dropout are incorporated to regulate the models' parameters.

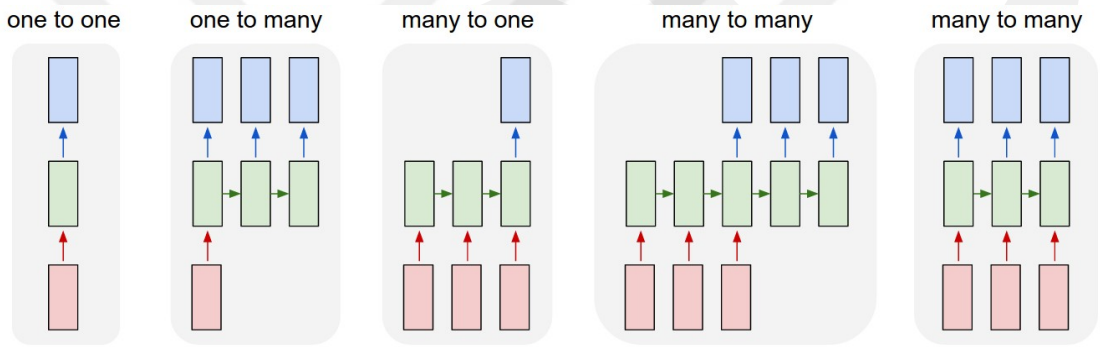


**Figure 3.10 Illustration of various types of data and tasks [64].**

On the other hand, transfer learning techniques are employed, involving the freezing of the convolutional base layers while retraining the upper layers, like the fully connected Multi-Layer Perceptron (MLP) layers. The frozen layers' weights are taken from models previously trained on the ImageNet dataset. After the flattening layer in the convolutional base, an MLP network with a single hidden layer containing 64 neurons is applied. The SoftMax activation function is chosen for the last layer due to the multi-class classification challenge. The Adam optimizer, with a learning rate of 0.01 and a categorical CE loss function, is used. A mini-batch size of 16 is selected, the training is capped at 50 epochs, and Model Checkpoint is deployed for saving model weights at each stage. The most effective epoch for the test set is identified through validation loss. The parameters of the models have been regularized using L2-norm regularization, dropout, and batch normalization. Furthermore, data augmentation techniques [65] like rotation,

flipping, and cropping have been applied to enhance the diversity and quantity of training images due to the limited number of samples available in the training set.

**Table 3.4 Configuration of LSTM and GRU layers for vehicle type classification.**

| Layer names | LSTM/GRU classifier |
|---|---|
| L1 | Masking () |
| L2 | LSTM/GRU (32) |
| L3 | Batch normalization |
| L4 | Dropout (0.01) |
| L5 | LSTM/GRU (16) |
| L6 | Batch normalization |
| L7 | Dropout (0.01) |
| L8 | LSTM/GRU (8) |
| L9 | Batch normalization |
| L10 | Dropout (0.01) |
| L11 | Dense (3) |

## 3.4 Results and Discussion

In this chapter, using a camera and a 3-D magnetic sensor node, data is collected on intermediate road traffic by taking 376 vehicle samples and identifying the types of vehicles. LSTM, GRU, SVM, and transfer learning algorithms are applied to the dataset for vehicle type classification. Transfer learning approaches include VGG16, VGG19, Xception, MobilNet, MobilNetV2, DenseNet121, DenseNet169, and DenseNet201. Models are trained on a training set and validated on a validation set, and then the model is also tested on the test set. The accuracies of LSTM and GRU models are obtained as 74.33% and 81.41%, respectively, with the following hyperparameters: a dropout rate of 0.2, a learning rate of 0.001, L2 norm regularization at 0.0001, and the Adam optimizer. Subsequently, transfer learning methods are applied, wherein the convolutional base layers of the models are frozen and the top layers are retrained. The same hyperparameters are used for this process: a dropout rate of 0.2, a learning rate of 0.001, L2 norm regularization set at 0.0001, and the optimizer chosen is Adam. The VGG16 model obtained the best results, with an accuracy of 92.03%. Moreover, the SVM model accuracy is obtained at 83.18% with the following optimum hyperparameters: C = 100,

gamma = 0.001, kernel = sigmoid. Table 3.5 shows all the performance results we obtained in this chapter for vehicle type classification.

**Table 3.5 Performance results of the machine learning and deep learning classifiers on three-dimensional vehicle type classification.**

| Classifier | Accuracy (%) | Precision (%) | Recall (%) | F1-measure (%) |
|---|---|---|---|---|
| LSTM | 74.33 | 85.83 | **84.17** | 76.63 |
| DenseNet201 | 78.76 | 75.40 | 47.27 | 76.10 |
| DenseNet169 | 79.64 | 81.76 | 69.62 | 80.15 |
| GRU | 81.41 | 84.35 | 76.28 | 82.29 |
| DenseNet121 | 82.30 | 84.33 | 82.93 | 71.79 |
| MobileNetV2 | 83.18 | 81.55 | 61.18 | 82.19 |
| SVM | 83.18 | 82.82 | 64.99 | 82.39 |
| Xception | 84.07 | 76.84 | 49.62 | 79.33 |
| VGG19 | 88.49 | 82.14 | 63.53 | 85.18 |
| MobileNet | 90.26 | 83.01 | 66.29 | 86.48 |
| VGG16 | **92.03** | **92.67** | 70.00 | **88.94** |
| Ensemble | **92.92** | **94.35** | **91.08** | **93.42** |

In the analysis of 3-D vehicle type classification using DL techniques, the results indicate that the choice of model has a significant impact on the classification outcomes, particularly when discriminating between different vehicle types. It has been observed that transfer learning models like VGG16, when fine-tuned, excel at distinguishing light and medium vehicles with notable precision, yet they underperform in classifying heavy vehicles, as illustrated in Figure 3.11(a). On the other hand, LSTM and GRU models, despite not securing the top spot for overall accuracy, exhibit superior capability in identifying medium and heavy vehicles, as depicted in Figures 3.11(b) and 3.11(c). This suggests their enhanced ability to capture complex patterns and temporal dependencies that are characteristic of these vehicle categories, a feature that is particularly beneficial in situations where accurate detection of larger vehicles is paramount for safety and compliance reasons.

Taking a comprehensive view of the performance metrics, the ensemble method employing custom soft voting emerges as a standout strategy. This method synergizes the strengths of the individual models, combining VGG16's proficiency with light and

medium vehicles and LSTM/GRU's effectiveness with medium and heavy ones to deliver superior classification performance. In this study, the ensemble method employs custom-weighted soft voting, with higher weights assigned to VGG16 for light and medium vehicle classes and elevated weights for LSTM and GRU for medium and heavy vehicle classes. The result is an impressive overall accuracy rate of 92.92% and an F1-measure of 93.42%, as shown in Figure 3.11(d). This nuanced application of weights within the soft voting framework allows for precise differentiation across all vehicle categories, as shown in Figure 3.11(d), which shows only 8 misclassified samples, underscoring the potency of ensemble methods in addressing complex classification tasks.

| | | Predicted | | | | |
|---|---|---|---|---|---|---|
| | | L | M | H | T | ACC (%) |
| Actual | Light (L) | 14 | 0 | 14 | 14 | 100 |
| | Medium (M) | 0 | 89 | 0 | 89 | 100 |
| | Heavy (H) | 1 | 8 | 1 | 10 | 10 |
| | Total (T) | 15 | 97 | 1 | 113 | 92.03 |

(a) Confusion matrix of the VGG16 method.

| | | Predicted | | | | |
|---|---|---|---|---|---|---|
| | | L | M | H | T | ACC (%) |
| Actual | Light (L) | 13 | 1 | 0 | 14 | 92.85 |
| | Medium (M) | 16 | 62 | 11 | 89 | 69.66 |
| | Heavy (H) | 0 | 1 | 9 | 10 | 90 |
| | Total (T) | 29 | 64 | 20 | 113 | 74.33 |

(b) Confusion matrix of the LSTM method.

| | | Predicted | | | | |
|---|---|---|---|---|---|---|
| | | L | M | H | T | ACC (%) |
| Actual | Light (L) | 12 | 2 | 0 | 14 | 85.71 |
| | Medium (M) | 9 | 74 | 6 | 89 | 83.14 |
| | Heavy (H) | 0 | 4 | 6 | 10 | 60 |
| | Total (T) | 21 | 80 | 12 | 113 | 81.41 |

(c) Confusion matrix of the GRU method.

| | | Predicted | | | | |
|---|---|---|---|---|---|---|
| | | L | M | H | T | ACC (%) |
| Actual | Light (L) | 14 | 0 | 0 | 14 | 100 |
| | Medium (M) | 0 | 83 | 6 | 89 | 93.25 |
| | Heavy (H) | 0 | 2 | 8 | 10 | 80 |
| | Total (T) | 14 | 85 | 16 | 113 | 92.92 |

(d) Confusion matrix of the soft voting ensemble method.

**Figure 3.11 Performance matrix of VGG16, LSTM, GRU, and custom ensemble classification methods, respectively.**

The loss history of the LSTM, GRU, and VGG16 models throughout training and validation, shown in Figures 3.12, 3.13, and 3.14, highlights the intricate dynamics involved in model training. The figure illustrates that the optimal loss rates for the validation sets are 1.1445, 0.3642, and 0.1970, achieved after 33, 162, and 200 iterations on the VGG16, LSTM, and GRU models, respectively. The VGG16 model exhibits fast convergence towards a reduced loss value, suggesting that it is effectively learning information from the training data. A sharp decrease in the loss value, particularly during the early stages of training, indicates that the model's parameters are being modified considerably to minimize the prediction error. If the validation loss exhibits a similar pattern, it would suggest that the model is not just overfitting to the training data but also effectively generalizing to new, unseen data. The LSTM model appears to have a more

gradual reduction in loss over a larger number of training steps. This could indicate that the LSTM, with its ability to capture long-term dependencies, is slowly assimilating the patterns in the sequential data. The GRU model, known for its simpler structure compared to LSTM, seems to be showing an interesting pattern where the validation loss is lower than the training loss. This phenomenon may occur when dropout or other regularization methods are used during the training phase. The decrease in validation loss suggests that the GRU model has a higher capacity to generalize to unfamiliar data, possibly attributed to its capability to prevent overfitting by using regularization approaches. Overall, the loss history figure is essential for comprehending the learning dynamics of different models. It helps in evaluating the convergence of the model, identifying problems of overfitting or underfitting, and providing guidance for additional enhancements to the model.

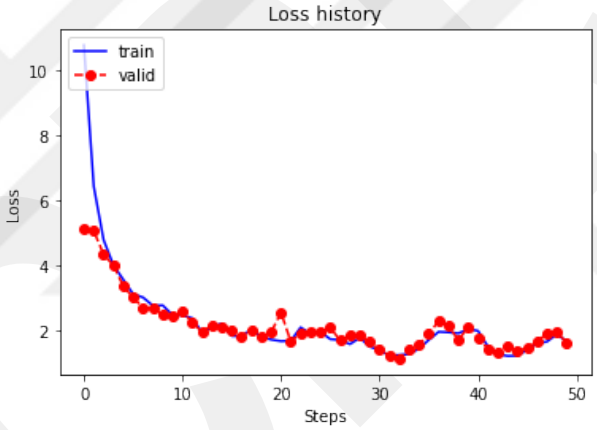


**Figure 3.12 The loss history of VGG16 model on training and validation steps.**

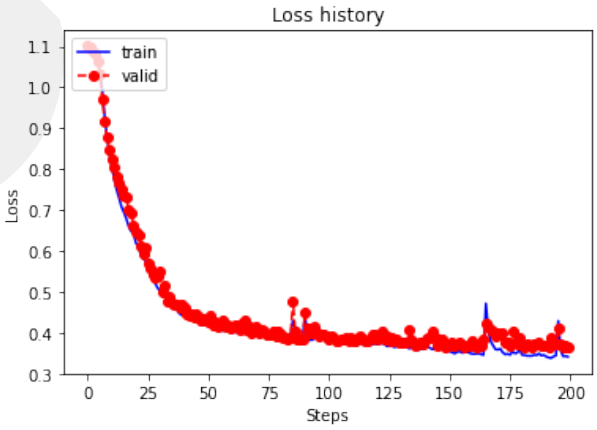


**Figure 3.13 The loss history of LSTM model on training and validation steps.**

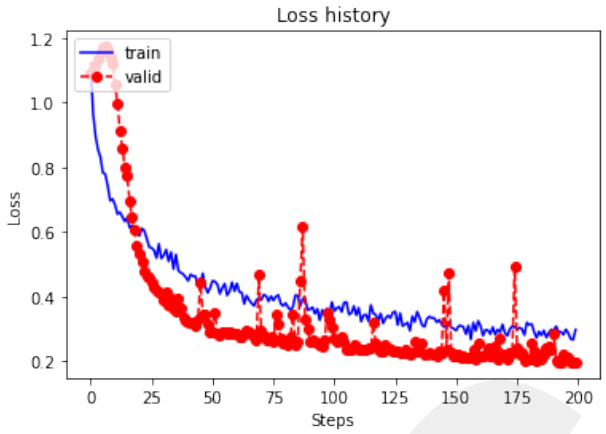


**Figure 3.14 The loss history of GRU model on training and validation steps.**

To summarize, the findings of this study confirm that vehicle type classification is a complex task requiring various methodological approaches. Transforming vehicle signals into 2-D images and employing transfer learning methods have proven to enhance performance scores. At the same time, LSTM and GRU methods have demonstrated the ability to accurately classify even the most challenging samples, albeit with a generally lower performance rate. The study indicates that while individual models each have their strengths, integrating their predictions through a soft voting ensemble method can create a more reliable and precise classification system. The insights gained from this study could lead to the development of more advanced ensemble strategies, potentially including a wider array of models and techniques, to further improve classification performance.

# Chapter 4

# An efficient network intrusion detection approach based on logistic regression model and parallel artificial bee colony algorithm

## 4.1 Introduction

In recent years, the number of people and applications using the internet has expanded dramatically, largely due to the development of smart technology. According to Data Reportal, which collects data about internet usage worldwide, approximately one million new internet users are added each day, with the total number of internet users increasing by 13% in 2020 compared to the previous year [66]. The increased usage of the internet has also brought numerous security challenges. Cybercrime and threat activities have become a critical concern, underscoring the growing importance of cybersecurity. SonicWall reported that around 4.8 trillion intrusion attempts occurred in 2020, representing a 20% increase from the previous year [67]. These intrusion attempts aim to penetrate information systems to steal or compromise sensitive data. To mitigate security vulnerabilities, technologies like firewalls, data encryption, and user authentication methods are employed. While these security measures are effective against a wide range of cyber threats, they fall short in conducting in-depth packet analysis and, consequently, may not detect all attacks or types of attacks. Therefore, as a result of these concerns, NIDS have been created to compensate for the deficiencies of the security methods, which monitor the network continually for malicious attacks and warn users when intrusions or attacks occur. NIDS are typically divided into two categories:

Signature-based and anomaly-based. Signature-based systems rely on a database of known malware signatures and can be less effective due to the ever-increasing variety of attacks; attackers can easily circumvent these systems with slight modifications to their methods. Moreover, as signature databases grow, these systems can become slower due to the continuous need to update and maintain the database. In contrast, anomaly-based detection systems establish patterns of normal behavior without relying on signatures and identify threats using the learned model. Thanks to their underlying ML algorithms, these models can conduct more in-depth data analysis.

Anomaly-based detection systems employ a range of ML methods, such as rule mining, classification, clustering, and DL algorithms, to protect network security by detecting intrusions and attacks with high accuracy and f1-measure. However, ML methods on their own are not without challenges; they often require additional data preprocessing steps guided by human expertise to address issues, which requires expert input, and they can struggle with issues such as anomaly detection, the significant impact of errors, discrepancies between results and their interpretation, variability in network traffic, the diversity of attack types, and difficulties in data evaluation [68]. Recently, it was shown that a variety of critical issues can be handled, such as massive network traffic, diverse data distribution, and continuously changing environmental circumstances, by integrating the ABC approach with ML methods [69]. To this end, the ABC algorithms offer several advantages: (i) they require less prior knowledge and expert intervention, enabling classification without specific data preprocessing [70]; (ii) hybridization with ML techniques enhances model performance [68]; (iii) ABC is less dependent on predefined labels in the dataset [71]; and (iv) it is inherently distributed, performing efficiently in parallel and distributed computing settings [72]. This study proposes a novel network anomaly detection approach using LR, renowned for its uncomplicated design, swift processing in real-time scenarios [73], and high operational efficiency. To address LR's inclination towards suboptimal local solutions, training is conducted using the ABC [74] algorithm. This algorithm, inspired by natural phenomena, emulates the foraging behavior of honeybees, leveraging the principles of swarm intelligence.

To the best of our knowledge, this study builds the first network anomaly detection approach that utilizes the LR model and ABC algorithm together. The ABC algorithm is effective in dealing with multimodal and complex, high-dimensional problems [75], [76]. It possesses a harmonious blend of exploration and exploitation capabilities, making it a

suitable choice for anomaly-based NIDS. Additionally, the computational time of the suggested solution is decreased via the use of parallel computing techniques, and the Bayesian hyperparameter optimization technique is used to optimize ML algorithms' hyperparameters. The proposed approach is evaluated using two publicly available network datasets: UNSW-NB15 and NSL-KDD. The performance of the proposed model is evaluated with state-of-the-art ML and DL models, such as DT, Linear Discriminant Analysis (LDA), LR, MLP, RF, SVM, XGBoost, DNN, LSTM, and GRU. Comparative experiments on the UNSW-NB15 and NSL-KDD datasets show that the proposed model outperforms other methods in accuracy, False Positive Rate (FPR), and F1-measure for UNSW-NB15, as well as in accuracy, False Negative Rate (FNR), and F1-measure for NSL-KDD, while reducing the training time. The proposed model achieves an accuracy of 88.25% on the UNSW-NB15 dataset and 90.11% on the NSL-KDD dataset, and F1-measures of 88.26% and 90.15%, respectively. Additionally, thanks to GPU parallelization, the proposed model's training time was approximately 4.45 times faster than the CPU version of the LR-ABC approach, indicating a significant improvement in execution speed. Overall, the major contributions of this chapter can be summarized as follows:

- This chapter proposes an efficient approach based on an LR-ABC algorithm for NIDS.
- To overcome the high computational time of the standard LR-ABC models, an efficient model has been developed based on CPU and GPU parallelization techniques to significantly reduce training time.
- The performance of the proposed approach outperforms the state-of-the-art ML and DL models in terms of accuracy, FPR, FNR, and F1-measure.
- Comparative performance evaluations are based on the publicly available UNSW-NB15 and NSL-KDD datasets, which are among the most comprehensive available datasets. The high performance of the proposed approach shows that the proposed model is reliable and robust to detect various attack types, and it provides a scalable solution for adapting to the dynamic and evolving landscape of cybersecurity threats.

- The Bayesian hyperparameter optimization method has been utilized to automatically optimize the hyperparameters of the proposed LR-ABC approach and state-of-the art machine learning and deep learning methods.

This chapter is organized as follows: Section 4.2 provides an overview of the current ML-based NIDS. Section 4.3 describes evaluation metrics, available datasets, preprocessing steps, and the hyperparameter optimization method. In Section 4.4, the proposed LR-ABC approach and the parallel computing method are explained. Section 4.5 outlines the experimental steps. Section 4.6 presents the performance results of the proposed LR-ABC and other classification methods.

## 4.2 Related Work

In recent years, attackers have been upgrading themselves and the software that they use and inventing new malicious activities. Until now, different ML-based NIDS have been developed. Anomaly-based NIDS are favored for their ability to identify novel attack types, unlike signature-based systems. Due to the automated nature of ML techniques, they are able to develop a variety of models without the strong involvement of human skills [77], which is sometimes a constraint and costly. For this purpose, many studies aim to increase the performance of anomaly-based NIDS for different types of cybersecurity attacks.

Hajisalem et al. [78] suggest a hybrid method for anomaly-based NIDS that combines the ABC and Artificial Fish Swarm algorithms. This hybrid method generates rules through the use of fuzzy C-means clustering and correlation-based feature selection techniques. They generate if-then rules using the CART technique to distinguish normal and anomalous records. Qureshi et al. [79] suggest a NIDS that utilizes a random neural network trained with the ABC algorithm to discover the ideal weights for the neurons, followed by a comparison to the classic gradient descent based RNN model. Mazini et al. [80] suggest a hybrid method that combines an ABC algorithm for feature selection to select the best subset of related features and an AdaBoost meta-algorithm for classification. Gu et al. [81] create a NIDS that uses SVM with the tabu-ABC for feature selection and parameter optimization at the same time. They adopted the tabu search algorithm to enhance the neighborhood search of ABC. It is utilized for reducing the feature size dimensions, and meanwhile, SVM parameters are optimized. Finally, the

dataset is utilized to train the SVM classifier model using the appropriate feature subset and hyperparameters. Rani et al. [82] use the ABC algorithm for the feature selection process and an RF classifier for classification tasks. Additionally, they demonstrate in other research why feature selection procedures result in overfitting and are unable to improve classification accuracy on NIDS [83]. In our previous studies [84], the ABC algorithm was applied to LR on e-mail spam filtering tasks and then evaluated on three public datasets. The proposed approach is compared with other ML algorithms. The suggested methodology outperforms other approaches in terms of classification accuracy and FPR. The proposed model demonstrates a high degree of effectiveness on unbalanced and nonlinear spam datasets. However, the suggested method has a shortcoming as it requires more training time compared to the other methods.

On the other hand, several studies on NIDS have focused on different preprocessing steps and used individual classifiers such as DT [85], LDA [86], LR [87], MLP [88], and SVM [89] on the NIDS dataset. While these studies provide valuable insights into NIDS, none have explored the LR-ABC classification for anomaly-detection in NIDS. In this study, the proposed LR-ABC approach is compared to the state-of-the-art ML and DL algorithms, which include DT, LDA, LR, MLP, RF, SVM, XGBoost, DNN, LSTM, and GRU classifiers, using the UNSWNB15 and NSL-KDD datasets. Additionally, this study emphasizes that no cleaning process was applied to the datasets, and feature selection methods were not used. While some preprocessing methods could potentially increase accuracy, such enhancements are outside the scope of this study.

Overall, ML and metaheuristic methods have been widely used for NIDS. However, existing studies on NIDS usually suffer from low performance results such as accuracy, F1-measure, FPR, and FNR. Moreover, current studies generally do not use automatic parameter tuning techniques. To address these challenges, this chapter proposes a novel approach based on a LR model trained using a parallel ABC algorithm with a hyperparameter optimization technique. To overcome the high computational time of the LR-ABC models, an efficient LR-ABC model has been developed based on CPU and GPU parallelization techniques to significantly reduce training time. To the best of our knowledge, this chapter proposes the first anomaly-based NIDS approach that employs the parallel ABC as an LR learning algorithm.

## 4.3 Materials and Methods

**Table 4.1 Traditional confusion matrix.**

| | Predicted Anormal | Predicted Normal |
|---|---|---|
| **Actual Abnormal** | TP | FN |
| **Actual Normal** | FP | TN |

### 4.3.1 Evaluation metrics

Accuracy is an essential criterion for evaluating a model's overall performance. The major goal of the current research is to increase the accuracy of NIDS, but the accuracy criterion may not be adequate in NIDS, especially in anomaly detection. Therefore, in addition to the accuracy metric, F1-measure, FPR, and FNR metrics, training time is also used to evaluate the classification performance. The FPR measures the rate of normal traffic falsely detected as anomalies, while the FNR indicates the rate of actual anomalies mistakenly classified as normal. The F1-measure, the harmonic mean of recall and precision, reflects the model's sensitivity and robustness. These are important details to be examined in the NIDS. These performance metrics are given in equations (4.1), (4.2), (4.3), and (4.4), respectively. These metrics help to assess the performance of the model in several aspects. The traditional confusion matrix is shown in Table 4.1.

$$Accuracy\ (ACC) = \frac{TP + TN}{TP + FN + FP + FN} \tag{4.1}$$

$$False\ Positive\ Rate\ (FPR) = \frac{FP}{TN + FP} \tag{4.2}$$

$$False\ Negative\ Rate\ (FPR) = \frac{FN}{TP + FN} \tag{4.3}$$

$$F1 - measure\ (F1) = \frac{2 * TP}{2 * TP + FP + FN} \tag{4.4}$$

### 4.3.2 Datasets

The UNSW-NB15 [90] and NSL-KDD [91] are benchmark network traffic datasets well-known in NIDS research. The goal of dataset generation is to provide a robust and realistic dataset. The UNSWNB15 dataset provides training and testing datasets separately. This dataset provides both multiclass and binary class labels. The training set contains a total of 175,341 samples, of which 56,000 are labeled ‘‘normal’’ and 119,341 are labeled ‘‘abnormal’’. Similarly, the testing set consists of 82,332 samples, of which 37,000 are labeled ‘‘normal’’ and 45,332 are labeled ‘‘abnormal’’ traffic samples. The dataset contains 45 features and abnormal classes containing nine attack types, including backdoors, analysis, DoS, exploits, fuzzers, generic, reconnaissance, shell code, and worms.

**Table 4.2 Class distribution of UNSW-NB15 and NSL-KDD datasets.**

| Datasets | Class | Training Set | Test Set |
|---|---|---|---|
| UNSW-NB15 | Normal | 56.000 | 37.000 |
| | Fuzzers | 18.184 | 6.602 |
| | Analysis | 2.000 | 677 |
| | Backdoors | 1.746 | 583 |
| | Dos | 12.264 | 4.089 |
| | Exploits | 33.393 | 11.132 |
| | Generic | 40.000 | 18.871 |
| | Reconnaissance | 10.491 | 3.496 |
| | Shellcode | 1.131 | 378 |
| | Worms | 130 | 44 |
| | **Total** | **175.341** | **82.332** |
| NSL-KDD | Normal | 67.343 | 9.711 |
| | Dos | 45.927 | 7.458 |
| | Probe | 11.656 | 2.421 |
| | U2R | 52 | 200 |
| | R2L | 995 | 2.754 |
| | **Total** | **125.973** | **22.544** |

The NSL-KDD is divided into two parts: KDDTrain+ and KDDTest+. The training set has 125,973 samples, with 67,343 labeled ‘‘normal’’ and 58,630 labeled ‘‘abnormal’’, including 22 attack types, which are categorized into four attack classes. The test set has

22,544 samples, of which 9711 are labeled ‘‘normal’’, and 12,833 are labeled ‘‘abnormal’’, including 37 attack types, also grouped into four attack classes. The distribution of four classes is: denial of service (DoS) attacks, root-to-local attacks (R2L), user-to-root attacks (U2R), and probing attacks (Probe). The NSL-KDD dataset contains 41 features.

### 4.3.3 One hot encoding

ML algorithms consider the magnitude of numerical values as the importance or significance of features. In other words, based on the categorical values, it will consider the higher number more important or superior to a lower number. Therefore, on the UNSW-NB15 dataset, which has categorical features, one hot encoding technique is applied to transform categorical features into numeric values. For instance, the ‘state’ feature has nine categorical values: ‘FIN’, ‘INT’, ‘CON’, ‘ECO’, ‘REQ’, ‘RST’, ‘PAR’, ‘URN’, and ‘no’. These were turned into binary vectors using the one-hot encoding method as follows: [1, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 1, 0, 0, 0, 0, 0, 0, 0, 0], . .. , [0, 0, 0, 0, 0, 0, 0, 0, 0, 1]. The ‘service’, ‘state’, and ‘proto’ are the three categorical features in the UNSW-NB15, dataset and after encoding, the total number of features increases to 199. Similarly, the NSL-KDD dataset contains categorical features such as ‘protocol type’, ‘service’, and ‘flag’, which, after binary encoding, expand the total feature count to 122.

### 4.3.4 Data normalization

Normalization eliminates the influence of various scales across features, thereby reducing the time required to train the model. There are several normalization approaches. To choose the most suitable one, the dataset is analyzed for sparsity, a measure indicating how prevalent zeros are. This metric indicates that max-abs normalization strategies should be used before classification. This max-abs normalization technique scales and transforms each feature independently, ensuring that each feature in the training set has a maximum absolute value of 1.0 and does not center or shift the values, hence preserving any sparsity. So, the max-abs normalization methods are applied to scale the feature values into the numeric range between 0 and 1.

**Table 4.3 Hyperparameter ranges for classification methods for UNSW-NB15 and NSL-KDD datasets.**

| Classifier | Parameter | Lowest | Highest |
|---|---|---|---|
| DT | Min Samples Split | 2 | 100 |
| | Min Samples Leaf | 1 | 100 |
| DL | Batch Size | 8 | 64 |
| | Learning Rate | 10^-6 | 10^-2 |
| | Neuron1 | 1 | 128 |
| | Neuron2 | 1 | 128 |
| | Neuron3 | 1 | 128 |
| LDA | Shrinkage | 0 | 1 |
| LR | C | 10^-4 | 10^4 |
| LR-ABC | LB | -64 | 0 |
| | UB | 0 | 64 |
| | Evaluation Number | 10.000 | 160.000 |
| | Limit | 10 | 500 |
| | P | 10 | 100 |
| | MR | 0.02 | 0.5 |
| | L2 | 0 | 0.1 |
| MLP | Learning Rate | 10^-8 | 10^-1 |
| | Number of Hidden Units | 2 | 40 |
| | Batch Size | 1 | 1024 |
| | Number of Epochs | 1 | 50 |
| RF | Number of Trees | 1 | 200 |
| SVM | C | 0.001 | 1 |
| XGBoost | Eta | 0.1 | 1 |
| | Depth | 1 | 40 |

### 4.3.5 Bayesian hyperparameter optimization

In ML, there are numerous parameter optimization strategies that guarantee the model will achieve the best performance in the given space. For this reason, hyperparameter selection is a critical procedure during training the model. The main advantage of hyperparameter selection is that it is applicable to handling parameter tuning for many different models. The parameter tuning process has a strong impact on the performance or efficacy of a model, but this usually requires a large number of runs. This makes the tuning process time-consuming, which is the main disadvantage of hyperparameter optimization [92]. The second disadvantage is that determining the parameter value is still challenging. Bayesian hyperparameter optimization enables the search of a larger hyperparameter space. For each parameter, the method accepts an interval (i.e., min and max values) and can consider any value within that interval. Another advantage is that Bayesian hyperparameter optimization can be completed in a

matter of days for the same search space and computational resources, whereas standard techniques can take up to a year [93]. Table 4.3 shows the hyperparameter ranges for classification methods in this study.

## 4.4 Proposed LR-ABC Method

### 4.4.1 Artificial bee colony (ABC) algorithm

The ABC algorithm, developed by Karaboga [74], is an optimization technique that emulates the food-gathering habits of honeybees. The ABC algorithm consists of employed bees that are actively engaged in the process, onlooker bees that observe and make decisions based on their observations, and scout bees that seek out new opportunities. In this context, the location of a food supply represents a potential resolution, whereas the quantity of nectar available at the food supply indicates the caliber of that resolution. The algorithm's main objective is to determine the food supply that has the greatest amount of nectar. The stages of the algorithms are detailed in Algorithm 1 on Table 4.4.

Employed bees have the task of remembering the more advantageous areas surrounding food sources. They communicate the details about food quantities and location to the onlooker bees in the dance area. The onlooker bees then decide which source to visit by observing the dance of the employed bees. The ABC algorithm mimics this dance and the effective selection of food sources using a stochastic selection technique. This technique incorporates a positive reaction, meaning that if a food source is rich in nectar, it attracts more onlooker bees to that particular source.

If a new solution discovered by an employed bee contains more nectar than the present one, the employed bee saves this new solution in its memory, replacing the old one. This occurs within the greedy selection phases of stages 4 and 8. Conversely, if the new solution isn't an improvement, the bee maintains the existing solution and adds one to its associated tally. These tallies keep track of how frequently a food source has been tapped and aid in determining when a source is exhausted during the scout bee stage. A food supply is deemed depleted when its counter exceeds a certain preset threshold. During each iteration of the fundamental algorithm, a maximum of one employed bee is allowed to transform into a scout bee. If multiple employed bees find their food sources

depleted, the algorithm selects the one with the highest tally value for this transformation. The bees abandon depleted sources, and scout bees then embark on a search for new, unexplored sources to take their place.

### 4.4.2 LR-ABC classification method

The LR model faces certain challenges with its gradient descent algorithm, such as the prerequisite of a continuous cost function. To overcome these limitations, the LR model is trained using the ABC method, which is a successful heuristic approach. In addition to making no assumptions about the function or parameter search space, the ABC algorithm can successfully search for both local and global solutions in the search space. The weights of the LR model trained with the ABC algorithm are similar to the locations of the food sources in the ABC algorithm. As a result, the method initially generates a population of starting weights and bias values. The bee stages aim to find the optimal weight set $\vec{w}_i$ and bias value that minimize the mean squared error at the model's output. The algorithm's steps are provided in Algorithm 2 on Table 4.5.

After a training set is given $\{(\vec{x}_1, y_1), \ldots, (\vec{x}_m, y_m)\}$ $y_i \in \{0, 1\}$ and $\vec{x}_1 \in R_n$, $1 \leqslant i \leqslant m$, LR-ABC classification model determines the class of the vector $\vec{x}_i$ by using equation (4.5), where $p_i$ is computed as seen in equation (4.6) and the function $\sigma$ corresponds to sigmoid function, which is given in equation (4.7). The LR-ABC method's objective is to find the weights $(\vec{w})$ that minimize the cost function, which is provided by equation (4.8). As can be seen from equation (4.8), the cost function includes the mean square error function with ridge regression (L2 regularization) used to avoid overfitting.

$$p_i = \begin{cases} 0, p_i < 0.5 \\ 1, p_i \geq 0.5 \end{cases} \tag{4.5}$$

$$p_i = \sigma\left(w_1 x_{i1} + w_2 x_{i2} + \cdots + w_n x_{n1} + b\right) \tag{4.6}$$

$$\sigma(\theta) = \frac{1}{1 + e^{-\theta}} \tag{4.7}$$

$$C(w) = \frac{1}{m}\sum_{i=1}^{m}(y_i - p_i)^2 + \frac{\lambda}{n}\sum_{j=1}^{n}(w_j)^2 \tag{4.8}$$

$$f_l = \frac{1}{1 + C_l} \tag{4.9}$$

The LR-ABC algorithm first randomly generates a total of P food source positions, as described in step 2 of Algorithm 2. Each food source's position corresponds to a weight vector. For each set of weight vectors, the LR-ABC model computes an output based on these weights and bias values. The fitness value of a solution is inversely proportional to the value returned by the cost function for that solution, as given in equation (4.9). Therefore, a solution with a higher cost value will have a lower fitness value. The fitness function directs the approach to the better locations in the search space, and Algorithm 3 on Table 4.6 demonstrates the calculation of the fitness value.

After evaluating the first group of bees, the approach continues to iterate through the phases of bees until the specified termination requirements are met. In the stage of the employed bee, a local search is undertaken around each current solution, leading to the generation of a new solution as outlined in the third step of Algorithm 1. In Algorithm 2, $\vec{\tau}$ is a vector that keeps track of the number of times each solution has failed to be improved, and in the scout bee phase, if there is a solution in this vector that is higher than the limit value, this solution is replaced with a new one.

## Table 4.4 Algorithm of the ABC.

**1:** Determine the number of food source ($\boldsymbol{P}$), maximum evaluation number ($\boldsymbol{MEN}$), limit and the number of parameters to be optimized ($\boldsymbol{n}$)

**2:** Randomly create the food source locations

for i ← *to* P:

for j ← *to* n:

$w_j^{min} \leftarrow lowerbound$

$w_j^{max} \leftarrow upperbound$

$\phi_{ij} \leftarrow random\,(0, 1)$

$w_{ij} = w_j^{min} + \phi_{ij}\, x\, (w_j^{max} - w_j^{min})$

**3:** Perform local searches around food source locations using employed bees

for i ← *to* P:

$\phi \leftarrow random$ [-1, 1]

k ← *random*Int [1, P] provided that $i \neq k$

j ← *random*Int [1, n]

$v_{ij} = x_{ij} + \phi$ x $(x_{ij} - x_{kj})$

**4:** Perform greedy selection

**5:** Calculate the fitness value for each food source; that is, evaluate the quality of each solution

for i ← *to* P:

$fi \leftarrow fitnessFunction(\vec{w})$

**6:** Calculate the probability value of each solution proportional to its quality

for i ← *to* P:

$\rho i \leftarrow 0.9 \times \frac{f_i}{\max(f)} + 1$

**7:** Onlooker bees select food sources by considering probability values

$i \leftarrow 0, j \leftarrow 0$

$while\ t < P$ :

$if\ random[0, 1) < pi$ :

$\phi \leftarrow random$ [−1, 1]

$k \leftarrow randomInt$ [1, $P$] $provided\ that\ i \neq k$

$j \leftarrow randomInt$ [1, $n$]

$v_{tj} = x_{ij} + \phi$ x $(x_{ij} - x_{kj})$

$t \leftarrow t + 1$

$i \leftarrow i + 1$

$if\ i \geqslant P$ :

$i \leftarrow 0$

**8:** Perform greedy selection

**9:** Scout bee phase: If there is an exhausted food source i then:

$for\ j \leftarrow 0\ to\ N$ :

$\phi_{ij} \leftarrow random[0, 1)$

$w_{ij} = w_j^{min} + \phi_{ij}\, x\, (w_j^{max} - w_j^{min})$

**Table 4.5 Algorithm of the proposed LR-ABC classification method.**

**Input:** Input matrix $\boldsymbol{X_{MxN}}$, target $\vec{\boldsymbol{y}}_{\boldsymbol{M}}$, number of food sources $\boldsymbol{P}$, position of the food sources $\boldsymbol{W_{PxD}}$, maximum evaluation number ($\boldsymbol{MEN}$), lower bound $\boldsymbol{lb}$, upper bound $\boldsymbol{ub}$

**Output:**

**1 :** $D \leftarrow N + 1$

**2 :** $W_{PxD} \leftarrow lb + rand(P, D) \times (ub - lb)$

**3 :** $W' \leftarrow W$

**4 :** $\overrightarrow{fit} \leftarrow CalculateFitness(W)$

**5 :** $\vec{\tau} \leftarrow zeros(P)$

**6 :** $evaluation\_number \leftarrow 0$

**7 : while** $evaluation\_number < MEN$ **do**

**8 :** Perform employed bee phase

**9 :** $\overrightarrow{sfit} \leftarrow CalculateFitness(W')$

**10:** $\overrightarrow{ind} \leftarrow \overrightarrow{sfit} > \overrightarrow{fit}$

**11:** $\overrightarrow{rand} \leftarrow \overrightarrow{sfit} > \overrightarrow{fit}$

**12:** $\vec{\tau}[\overrightarrow{ind}] \leftarrow 0$

**13:** $W[\overrightarrow{ind}] \leftarrow W'[\overrightarrow{ind}]$

**14:** $\overrightarrow{fit}\ [\overrightarrow{ind}] \leftarrow \overrightarrow{sfit}\ [\overrightarrow{ind}]$

**15:** $\vec{\tau}[\overrightarrow{rand}] \leftarrow \vec{\tau}[\overrightarrow{rand}] + 1$

**16:** Calculate probability values of all solutions

**17:** Perform onlooker bee phase

**18:** $\overrightarrow{sfit} \leftarrow CalculateFitness(W')$

**19:** **for** $i \leftarrow 1 : P$ **do**

**20:** $t \leftarrow \overrightarrow{tmpID}[i]$

**21:** **if** $\overrightarrow{sfit}\ [i] > \overrightarrow{fit}\ [t]$ **then**

**22:** $\vec{\tau}\ [t] \leftarrow 0$

**23:** $W[t, :] \leftarrow W'[i, :]$

**24:** $\overrightarrow{fit}\ [t] \leftarrow \overrightarrow{sfit}\ [i]$

**25:** **else**

**26:** $\vec{\tau}\ [t] \leftarrow \vec{\tau}\ [t] + 1$

**27:** **end if**

**28:** **end for**

**29:** Perform scout bee phase

**30:** Memorize best source

**31: end while**

**32:** Return global best solution

**Table 4.6 Algorithm of the calculation of the fitness function.**

procedure $CalculateFitness(\phi)$

1: $w \leftarrow \phi$ [:, 1 :]

2: $b \leftarrow \phi$ [:, 0]

3: $p \leftarrow \sigma\,(X.dot(w^T) + b)$

4: $f \leftarrow mean\,((\overrightarrow{y_M} - p)^2\,, axis = 0)$

5: $f \leftarrow \frac{1}{(f+1)}$

6: $evaluation\_number \leftarrow evaluation\_number + len(f)$

7: $return\ f$

### 4.4.3 Computation on GPU

The CPU version of the LR-ABC and state-of-the-art ML methods are implemented using the NumPy [94] library, which does not support GPU computations. NumPy is widely used in ML methods, and the Python community has built packages like scikit-learn on top of NumPy. With the rapid development of GPU technologies over the last few years, researchers have increasingly focused on parallel computing to accelerate algorithms. However, the CPU version of the LR-ABC method still has limitations due to its high computational time. Due to the large size of the training datasets (175,341 × 199 and 125,973 × 122 matrices for UNSW-NB15 and NSL-KDD, respectively), accelerating our model became imperative. Specifically, by harnessing GPU parallelization for vectorized loops, we significantly enhanced overall speed and efficiency. Additionally, we parallelized common array operations such as searching, comparing, addition, subtraction, and matrix multiplications on a GPU. We achieved this using CuPy [95], an open-source library that accelerates matrix operations using NVIDIA GPUs. CuPy is compatible with NumPy and enables full use of modern GPU capabilities through a NumPy-compatible interface. The improved parallel LR-ABC method was developed with the CuPy library. As a result, as shown in the performance results in Section 4.6, the training time has been dramatically reduced by approximately 4.5 times compared to the CPU version of LR-ABC. A detailed implementation of the GPU version can be accessed in the references [96], [97].

## 4.5 Experiments

In our experiment, eleven classification algorithms – DT, LDA, LR, MLP, RF, SVM, XGBoost, DNN, LSTM, GRU, and the proposed LR-ABC method – are evaluated using the publicly available UNSW-NB15 and NSL-KDD datasets. In the preprocessing steps for both datasets, the one hot encoding technique is applied to transform categorical features into numeric values. Max-abs normalization methods are used to map the numeric feature values into the range 0 to 1. The classification methods are implemented with a default parameter and using the Bayesian optimization technique, where for each classifier, the run count limit is set to 100. Table 4.3 shows the hyperparameter ranges for classification methods for the UNSW-NB15 and NSL-KDD datasets. Individual results in terms of accuracy, FPR, FNR, and F1-measure are obtained for each classifier. Since both the UNSW-NB15 and NSL-KDD datasets contain separate training and test sets, each model is trained on the training set and evaluated on the test set.

The architectures of the DNN, LSTM, and GRU models consist of an input layer, three hidden layers, and an output layer. The hidden layers are configured with 64, 64, and 16 neurons, respectively, each employing a ReLU activation function. The last layer utilizes a sigmoid activation function. To prevent the risk of overfitting, batch normalization is implemented after each layer. The training process utilizes the Adam optimizer with the binary CE loss function. Additionally, an early stopping mechanism stops model training if the validation loss does not improve after five epochs, at which point the best model weights are reinstated. The learning rate is set to its default value, and the models are trained with a mini-batch size of 32 for a maximum of 100 epochs.

All ML models are implemented using the Scikit-Learn library [53], while all DL models are implemented using Keras [54]. The proposed approach was developed in the Python programming language [52].

## 4.6 Results and Discussion

In this chapter, the proposed LR-ABC method, along with seven different ML methods (DT, LDA, LR, MLP, RF, SVM, and XGBoost) and three different DL methods (DNN, LSTM, and GRU), is experimented with two publicly available NIDS datasets for anomaly detection in network traffic. Each classifier is trained with a default parameter and with hyperparameter optimization using the Bayesian technique.

**Table 4.7 Performance results of the proposed and other classification methods with default parameters on NSL-KDD datasets.**

| Classifier | Accuracy (%) | FPR | FNR | F1-measure (%) |
|---|---|---|---|---|
| DT | 79.73 | 0.0489 | 0.3189 | 79.66 |
| LDA | 76.16 | 0.0678 | 0.3673 | 75.98 |
| LR | 75.39 | 0.0743 | 0.3759 | 75.20 |
| LR-ABC | 74.48 | 0.0728 | 0.3932 | 74.21 |
| MLP | **81.83** | 0.0783 | **0.2597** | **81.89** |
| RF | 81.23 | 0.0702 | 0.2765 | 81.25 |
| SVM | 76.29 | 0.0732 | 0.3610 | 76.14 |
| XGBoost | 78.69 | **0.0282** | 0.3529 | 78.49 |
| DNN | 78.54 | 0.0723 | 0.3220 | 78.50 |
| LSTM | 77.24 | 0.0333 | 0.3745 | 76.96 |
| GRU | 75.67 | 0.0390 | 0.3977 | 75.31 |

Performance results of the proposed and other classification methods with default parameters on NSL-KDD are shown in Table 4.7. The MLP classifier shows the highest accuracy, FNR, and F1-measures, with 81.83%, 0.2597, and 81.89%, respectively, suggesting it is well-suited to this dataset. In contrast, the proposed LR-ABC method has lower accuracy and F1-measure, indicating that the default parameters are not optimal for this model; it requires more careful tuning to match the dataset's characteristics, highlighting the necessity for hyperparameter optimization. Also, the LR model has one of the lowest accuracies and F1-measures, with 75.39% and 75.20%, respectively. This might be due to the linear nature of LR, which could struggle with the complex, non-linear relationships in the NSL-KDD data. The XGBoost classifier has a relatively low FPR, which is desirable in many security applications to avoid over-alerting. However,

its FNR is high, which could be more critical as it implies missing actual threats. Conversely, the MLP has a lower FNR compared to other models but compensates with a higher FPR, which might be more acceptable depending on the cost of false alarms versus missed detections in the application context. The better performance of tree-based methods (DT, RF, and XGBoost) suggests that the dataset has feature interactions and non-linear decision boundaries that these models can capture effectively. The observed performances suggest that while some models like MLP and XGBoost are quite adaptable to the NSL-KDD dataset with default parameters, others, including the proposed LR-ABC method, require careful tuning and consideration of the unique characteristics of the dataset.

**Table 4.8 Performance results of the proposed and other classification methods with optimum hyperparameters found by Bayesian optimization on NSL-KDD datasets.**

| Classifier | Accuracy (%) | FPR | FNR | F1-measure (%) |
|---|---|---|---|---|
| DT | 82.42 | 0.0310 | 0.2851 | 82.40 |
| LDA | 78.70 | **0.0303** | 0.3511 | 78.51 |
| LR | 75.67 | 0.0747 | 0.3707 | 75.49 |
| LR-ABC | **90.11** | 0.0756 | **0.1163** | **90.15** |
| MLP | 85.86 | 0.0778 | 0.1894 | 85.93 |
| RF | 84.02 | 0.0314 | 0.2569 | 84.03 |
| SVM | 76.53 | 0.0733 | 0.3567 | 76.40 |
| XGBoost | 81.46 | 0.0282 | 0.3042 | 81.39 |
| DNN | 81.75 | 0.0475 | 0.2845 | 81.74 |
| LSTM | 81.75 | 0.0374 | 0.2922 | 81.71 |
| GRU | 84.12 | 0.1398 | 0.1729 | 84.19 |

Table 4.8 shows the performance results of the proposed and other classification methods using Bayesian hyperparameter optimization on the NSL-KDD dataset. The LR-ABC classifiers show the highest accuracy, FNR, and F1-measures, with 90.11%, 0.1163, and 90.15%, respectively, with the following optimum hyperparameters: LB = −2, UB = 46, Evaluation Number = 48296, Limit = 196, P = 10, MR = 0.40421, L2 = 0.06280, and a threshold of 0.8. The improvement in LR-ABC's performance after optimization is particularly striking, suggesting that its default parameter setting is suboptimal and that it benefits significantly from the Bayesian optimization process. Generally, the

optimization process has led to a performance enhancement in most classifiers, as seen by comparing these results to those obtained with default parameters. This underscores the importance of tuning hyperparameters specific to the dataset and the model. The varying degrees of improvement across classifiers suggest that the NSL-KDD dataset may contain complex feature interactions that are better captured by models with sufficient flexibility and capacity, like LR-ABC and MLP.

**Table 4.9 Performance results of the proposed and other classification methods with default parameters on UNSW-NB15 datasets.**

| Classifier | Accuracy (%) | FPR | FNR | F1-measure (%) |
|---|---|---|---|---|
| DT | 86.17 | **0.2493** | 0.0475 | 85.94 |
| LDA | 80.89 | 0.4217 | 0.0026 | 79.76 |
| LR | 80.54 | 0.3995 | 0.0271 | 79.63 |
| LR-ABC | 80.97 | 0.4227 | **0.0005** | 79.82 |
| MLP | 84.94 | 0.3199 | 0.0123 | 84.42 |
| RF | 86.64 | 0.2713 | 0.0211 | 86.31 |
| SVM | 81.59 | 0.4050 | 0.0035 | 80.58 |
| XGBoost | 87.37 | 0.2609 | 0.0162 | 87.07 |
| DNN | 86.14 | 0.2844 | 0.0195 | 85.76 |
| LSTM | **87.52** | 0.2564 | 0.0173 | **87.23** |
| GRU | 83.60 | 0.3573 | 0.0060 | 82.89 |

Table 4.9 shows the results of the proposed and other classification methods with default parameters on the UNSW-NB15 dataset. The LSTM model achieves the highest accuracy and F1-measure with 87.52% and 87.23%, respectively, suggesting that its ability to process sequences in the data is highly beneficial for the UNSW-NB15 dataset. Conversely, the DT model, despite having the lowest FPR with a 0.2493, shows compromised accuracy and F1-measure, pointing towards a potential underfitting issue. The default parameters have yielded mixed outcomes, and classifiers such as LSTM and XGBoost have performed well, suggesting their default configurations are robust for the UNSW-NB15 dataset. The high FPR for LDA and LR might suggest that their default decision boundaries are too lenient, causing many negative instances to be classified as positive. The MLP and RF classifiers show a good balance between FPR and FNR, which is reflected in their strong F1 measures. This balance suggests that they are good

candidates for scenarios where both types of classification errors are equally undesirable. The varying FPR and FNR rates among the classifiers could also point to the diverse nature of the attacks represented in the dataset. Some attacks may be easier to detect, leading to low FNR for certain classifiers, while others may closely mimic normal behavior, leading to high FPR for other classifiers.

**Table 4.10 Performance results of the proposed and other classification methods with optimum hyperparameters found by Bayesian optimization on UNSW-NB15 datasets.**

| Classifier | Accuracy (%) | FPR | FNR | F1-measure (%) |
|---|---|---|---|---|
| DT | 86.81 | 0.2642 | 0.0237 | 86.5 |
| LDA | 80.91 | 0.4218 | **0.0023** | 79.77 |
| LR | 80.89 | 0.3892 | 0.0292 | 80.04 |
| LR-ABC | **88.25** | **0.1212** | 0.1143 | **88.26** |
| MLP | 83.37 | 0.3500 | 0.0162 | 82.72 |
| RF | 86.79 | 0.2679 | 0.0211 | 86.47 |
| SVM | 81.59 | 0.4050 | 0.0035 | 80.58 |
| XGBoost | 86.93 | 0.2650 | 0.0210 | 86.62 |
| DNN | 87.81 | 0.2492 | 0.0174 | 87.54 |
| LSTM | 85.86 | 0.0161 | 0.2946 | 85.45 |
| GRU | 84.64 | 0.3233 | 0.0150 | 84.10 |

Table 4.10 shows the performance results of the proposed and other classification methods with hyperparameter optimization found by the Bayesian technique on the UNSW-NB15 dataset. Hyperparameter ranges for classification methods and optimum parameters found by the Bayesian technique are shown in Table 4.11. For the UNSW-NB15 dataset, the proposed LR-ABC method achieved the highest accuracy (88.25%), F1-measure (87.86%), and lowest FPR (0.1212) with the following optimum hyperparameters: lower bound (LB) = −20, upper bound (UB) = 10, Evaluation Number = 77885, Limit = 141, population size (P) = 15, mutation rate (MR) = 0.0100, L2 Regularization (L2) = 2.8368, and a threshold of 0.8. The optimization of hyperparameters using the Bayesian technique has likely contributed to the classifier. For instance, the DT, LDA, LR, XGBoost, DNN, and GRU classifiers showed slight improvements in accuracy. MLP and LSTM's accuracy and F1-measure decreased after

Bayesian optimization. The proposed LR-ABC method showed the most dramatic improvement with optimization, suggesting that its performance is highly dependent on the right set of hyperparameters. In general, some classifiers like MLP and DT performed well with default parameters, suggesting a natural compatibility with the dataset. In contrast, the LR-ABC method required optimization to achieve its best performance, highlighting the importance of tailored model configuration for anomaly detection tasks.

**Table 4.11 Optimum parameters found by Bayesian optimization on UNSW-NB15 and NSL-KDD Datasets.**

| Classifier | Parameter | UNSW-NB15 | NSL-KDD |
|---|---|---|---|
| DT | Min Samples Split | 99 | 19 |
| | Min Samples Leaf | 42 | 3 |
| DL | Batch Size | 55 | 60 |
| | Learning Rate | 10.0052 | 0.0040 |
| | Neuron1 | 37 | 9 |
| | Neuron2 | 29 | 84 |
| | Neuron3 | 127 | 50 |
| LSTM | Batch Size | 31 | 20 |
| | Learning Rate | 0.0028 | 0.0013 |
| | Neuron1 | 115 | 73 |
| | Neuron2 | 74 | 38 |
| | Neuron3 | 118 | 94 |
| GRU | Batch Size | 44 | 61 |
| | Learning Rate | 0.0062 | 0.0036 |
| | Neuron1 | 28 | 95 |
| | Neuron2 | 122 | 94 |
| | Neuron3 | 46 | 73 |
| LDA | Shrinkage | 5.26e-05 | 0.8579 |
| LR | C | 50.000 | 48.214,53 |
| LR-ABC | LB | -20 | -2 |
| | UB | 10 | 46 |
| | Evaluation Number | 77.885 | 48.296 |
| | Limit | 141 | 196 |
| | P | 15 | 10 |
| | MR | 0.0100 | 0.40421 |
| | L2 | 2.836e-05 | 0.0628 |
| MLP | Learning Rate | 0.0714 | 0.7574 |
| | Number of Hidden Units | 22 | 13 |
| | Batch Size | 527 | 603 |
| | Number of Epochs | 10 | 47 |
| RF | Number of Trees | 173 | 4 |
| SVM | C | 0.9965 | 0.7331 |
| XGBoost | Eta | 0.2305 | 0.2148 |
| | Depth | 37 | 4 |

Bayesian optimization has notably improved the performance of the LR-ABC method, as evidenced by the significant increase in accuracy and F1-measure relative to its baseline performance with default parameters on the NSL-KDD and UNSW-NB15 datasets. The observed trade-offs between FPR and FNR across various classifiers underscore the critical importance of model selection in accordance with the specific error costs pertinent to the application domain. The NSL-KDD dataset, characterized by its unique challenges, appears to be well addressed by sophisticated models, particularly when they are fine-tuned. Conversely, while fine-tuning did not uniformly benefit all classifiers on the UNSW-NB15 dataset, it notably promoted the performance of the LR-ABC models, thereby underscoring their potential as formidable contenders for network anomaly detection tasks. The study underscores the need for a carefully calibrated balance between false alarms and missed detections, tailored to the unique requirements and constraints of network security.

In the UNSW-NB15 dataset, as depicted in Figure 4.5, the LR-ABC method demonstrates variable accuracy across different attack types. It performs exceptionally well on backdoor, DoS, reconnaissance, and analysis types, achieving accuracies of 91% or higher. However, it notably falls short in correctly identifying 'generic' attacks, with an accuracy of only 60%. As presented in Figure 4.6, concerning the NSL-KDD dataset, the model shows strong performance on DoS and Probe attacks with accuracies of 94% and 92%, respectively. Yet, there is a significant drop in accuracy for R2L and U2R attack types, down to 73% and 69%, highlighting areas that require improvement. The LR-ABC model's varied performance across both datasets suggests that it is adept at detecting certain attack types, such as backdoors and DoS, which likely have more distinct and recognizable patterns. The diminished ability to identify 'Fuzzers' attacks in UNSW-NB15 and 'U2R' and 'R2L' attacks in NSL-KDD, which are less represented in the datasets, indicates that these attack types may be more complex or less defined, posing challenges for the model's classification capabilities. The sparse representation of 'U2R' attacks, in particular, may contribute to the classifiers' difficulties in accurate identification.

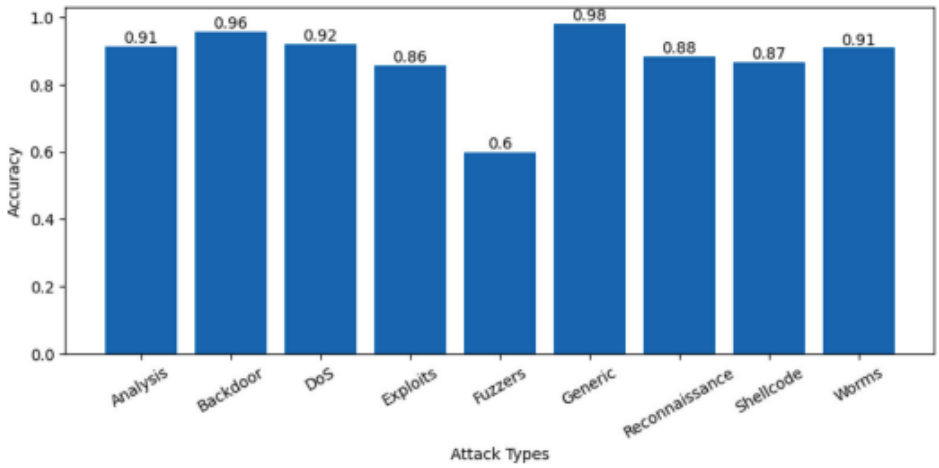


**Figure 4.1 The proposed LR-ABC method's accuracy for each different attack type on the UNSW-NB15 dataset.**

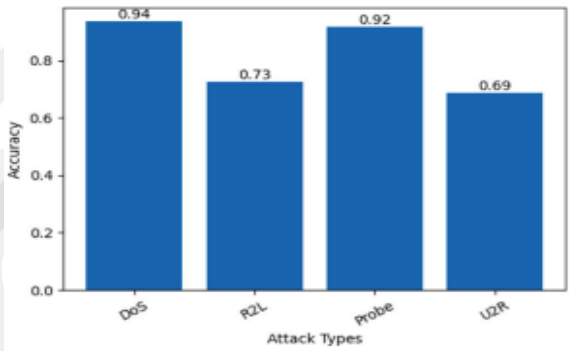


**Figure 4.2 The proposed LR-ABC method's accuracy for each different attack type on the NSL-KDD dataset.**

In practical applications, particularly within the domain of network security, training time is a critical measure of an algorithm's efficiency, along with performance metrics. The LR-ABC method demonstrates this, utilizing the NumPy and CuPy libraries to optimize training time on both CPU and GPU platforms. Its GPU implementation significantly cuts the average training time down to 121.56 seconds, outperforming the CPU version and rivaling classifiers such as XGBoost. Table 4.12 displays the average training times for each classifier across ten iterations. The LR-ABC model's swift training capability bolsters its suitability for dynamic environments that require timely model updates. While the precision of threat detection remains paramount, the ability to quickly train and retrain models is equally crucial in cybersecurity, allowing for rapid adaptation to new and emerging threats.

**Table 4.12 The training time of each classifier in seconds on UNSW-NB15 dataset.**

| Classifier | Best | Worst | Mean | Std. |
|---|---|---|---|---|
| DT | 3.04 | 3.11 | 3.09 | 0.88 |
| LDA | 7.57 | 7.65 | 7.60 | 0.02 |
| LR | 6.69 | 6.75 | 6.73 | 0.02 |
| LR-ABC (CPU) | 541.61 | 542.51 | 541.91 | 0.25 |
| LR-ABC (GPU) | 121.38 | 122.41 | 121.56 | 0.28 |
| MLP | 436.60 | 790.39 | 534.09 | 109.93 |
| RF | 165.30 | 168.69 | 166.83 | 1.12 |
| SVM | 1437.91 | 1559.12 | 1505.68 | 53.15 |
| XGBoost | 120.21 | 120.96 | 120.47 | 0.25 |
| DNN | 129.91 | 391.74 | 246.37 | 82.38 |
| LSTM | 827.90 | 2322.50 | 1637.03 | 517.88 |
| GRU | 384.75 | 1130.84 | 716.75 | 309.28 |

# Chapter 5

# Conclusions and Future Prospects

## 5.1 Conclusions

This thesis contributes to the fields of ITS and NIDS within the broader realm of the IoT. It addresses challenges and proposes innovative solutions. In ITS, the escalating need for efficient and effective vehicle type classification is driven by population growth. Effective vehicle type classification is essential for improving traffic management and congestion control, facilitating long-term infrastructure planning, enhancing public transportation and urban planning, ensuring environmental monitoring, and promoting road safety. These contributes significantly enhance the overall quality of urban life and form the backbone of modern ITS solutions.

This research addresses this need by developing an affordable, battery-operated 3-D magnetic sensor for accurate vehicle type classification. The integration of a DNN classifier, enhanced with hyperparameter optimization, feature selection, and extraction methods, is an important advancement to vehicle type classification. This approach achieves a high accuracy of 91.15% and an f-measure of 91.50%. Furthermore, in the second phase of research, DL techniques are combined with a custom soft voting ensemble method, which achieves even higher accuracy of 92.92% and an f-measure of 93.42%. In conclusion, this thesis has made significant contributions to the field of vehicle type classification. It has been demonstrated that while individual models possess inherent strengths, the confluence of these models through sophisticated ensemble methods can yield a classification system of remarkable accuracy and reliability. This advancement in vehicle type classification technology is important for the sustainable development of transportation infrastructure, particularly on side roads, enhancing comfort and road safety, and paving the way for the development of smart cities. It is

marking a pioneering approach in terms of energy-efficient traffic monitoring systems, addressing the growing environmental concerns and operational costs associated with ITS.

Meanwhile, NIDS are fundamentally important to the security of IoT devices and networks. The IoT domain, encompassing diverse internet-connected entities such as sensors, actuators, and a variety of smart devices, generates substantial data volumes. As IoT devices and networks proliferate, securing data and network infrastructure has become a critical concern. Protecting these devices from cyber threats is critical. NIDS are one of the keys to detecting and addressing potential security vulnerabilities within IoT networks and ensuring the integrity and reliability of data transmitted across IoT networks. These systems provide critical capabilities in monitoring, protecting against threats, ensuring data integrity, and complying with regulatory standards, which are all essential for the successful deployment of IoT technologies. They underscore the importance of cybersecurity in the IoT era. This research addresses inherent challenges in ML-based NIDS, such as handling high dimensionality, class imbalance, and the dynamic nature of network threats.

To effectively addressing these challenges, an efficient LR-ABC algorithm is proposed. The LR-ABC model demonstrates superior performance in network anomaly detection, outperforming existing ML and DL models with high accuracy and F1-measures on benchmark UNSW-NB15 and NSL-KDD datasets. The development of CPU and GPU versions of this model marks a significant improvement in training times, enabling rapid adaptation to evolving cybersecurity threats, which is essential for maintaining the security of IoT networks. The findings from this research are pivotal in enhancing both ITS and NIDS within the IoT context. This thesis not only contributes significantly to the advancement of ITS and NIDS but also lays the groundwork for future research.

By leveraging IoT technologies, this research paves the way for developing more robust, efficient, and adaptable systems, crucial for the rapidly evolving digital landscape and the growing demands of modern urban environments and cybersecurity.

## 5.2 Societal Impact and Contribution to Global Sustainability

The United Nations has unveiled the 2030 Agenda for Sustainable Development, a global initiative aimed at eradicating poverty, safeguarding the environment, and promoting peace and prosperity for everyone. At the center of this agenda are the 17 Sustainable Development Goals (SDGs), which form a collective blueprint, both in the present and for the future. These interconnected goals provide solutions to worldwide issues such as poverty, inequality, climate change, environmental decay, and social justice.

The disciplines of electrical and computer engineering are ideally positioned to make significant contributions toward these goals. Through innovative technologies and solutions, it has the potential to drive progress in various areas, from climate action to sustainable cities, equitable healthcare, and beyond. Especially, ML presents unparalleled opportunities to accelerate sustainable development with its ability to analyze vast data, derive insights, predict outcomes, and automate complex processes. This thesis contributes to the field within the context of the IoT, which has notable societal impacts and contributions to global sustainability. The research aligns with several SDGs, including Industry, Innovation, and Infrastructure (SDG 9), Sustainable Cities and Communities (SDG 11), Climate Action (SDG 13), Peace, Justice, and Strong Institutions (SDG 16), and Partnerships for the Goals (SDG 17).

In the realm of sustainable development, the creation of resilient infrastructure, promotion of inclusive and sustainable industrialization, and support for innovation are crucial objectives, as highlighted by the 9th SDG. The development of algorithms for vehicle type classification and the implementation of a 3-D magnetic sensor align with these objectives, enhancing smarter transportation systems and contributing to infrastructure innovation. Furthermore, the 11th SDG emphasizes transforming cities and human settlements into inclusive, safe, resilient, and sustainable environments. This goal is supported by the advancements in vehicle type classification technology, which, through the integration of ML, aid in effective traffic management and urban planning. This not only mitigates traffic congestion but also improves road safety, thereby contributing to sustainable urban development. Addressing climate change and its impacts, a core focus of the 13th SDG, is also a significant aspect of this research.

Implementing ML in traffic monitoring and management contributes not just to improved transportation efficiency but also significantly aids in the reduction of greenhouse gas emissions. This is in line with worldwide initiatives aimed at combating climate change. The 16th SDG underscores the importance of reducing cyber violence and increasing the accountability and transparency of digital systems. In this context, NIDS are critical. They ensure the protection of essential data and information, safeguarding public access and fundamental freedoms. Additionally, the 17th SDG brings to light the importance of global partnerships in achieving sustainable development. The interdisciplinary approach of this research, which encompasses computer science, artificial intelligence, transportation, and cybersecurity, exemplifies the necessity of collaborative efforts across various sectors and fields. Such collaborations are pivotal in driving innovation and achieving the SDGs.

This thesis demonstrates how advancements in electrical and computer engineering, especially in ITS and network security, extend beyond technological achievements; they are vital for promoting sustainable development and significantly contribute to the overarching goals of the SDGs.

## 5.3 Future Prospects

Building upon the significant contributions of this thesis to ITS and NIDS, several promising avenues for future research emerge. These prospects aim to extend the current findings and address emerging challenges in these rapidly evolving fields.

In ITS, expanding the data set and diversifying the type of vehicle can be effective in increasing the robustness and applicability of classification models. Another exciting possibility in ITS lies in the exploration of advanced model architectures for vehicle type classification. The use of cutting-edge architectures like EfficientNet, Inception, and NasNet, especially in conjunction with transfer learning techniques, can promise significant improvements in classification accuracy and efficiency. Transfer learning methods, known for their high performance in image and pattern recognition tasks, can provide significant benefits when applied to vehicle type classification using 3-D magnetic sensor data. The implementation and testing of developed ITS models in real-world scenarios are very important. This includes deploying 3-D magnetic sensor systems in real traffic environments to verify their effectiveness and reliability under changing

traffic conditions and environmental factors. Real-world tests will provide invaluable information about the practical challenges and performance of the systems and will guide further improvements.

As the IoT continues to grow, developing NIDS to protect IoT networks will become increasingly important. Future research should focus on developing NIDS that are not only effective in detecting a wide variety of attacks but also capable of rapidly adapting to new and evolving threats. This involves exploring advanced ML techniques to increase the accuracy and speed of network anomaly detection. Moreover, there is a need for models to achieve high performance not only in binary classification but also in multi-class classification models. Future work should focus on developing models that improve the detection and classification of various network attacks. This is particularly important for ensuring comprehensive security in complex network environments. The direct integration of developed NIDS models into IoT devices presents a promising research direction. This integration, allowing for more decentralized and efficient network monitoring and attack detection, will potentially provide faster response times and less network load.

Finally, both ITS and NIDS will greatly benefit from collaborative and interdisciplinary research efforts. Collaboration between traffic engineers, cybersecurity experts, data scientists, and urban planners can lead to more holistic solutions that address the multifaceted challenges in these fields. Additionally, interdisciplinary research can facilitate the development of innovative approaches that benefit from insights and techniques from different fields. As a result, the future of ITS and NIDS research is vibrant and full of potential. By building on the foundations laid by this thesis and exploring these future expectations, significant advancements can be made that contribute to smarter, safer, and more secure transportation systems and network environments.

# BIBLIOGRAPHY


[1] M. A. Jabraeil Jamali, B. Bahrami, A. Heidari, P. Allahverdizadeh, and F. Norouzi, "Towards the Internet of Things," 2020, doi: 10.1007/978-3-030-18468-1.

[2] M. Won, S. Sahu, and K. J. Park, "DeepWiTraffic: Low cost WiFi-based traffic monitoring system using deep learning," *Proceedings - 2019 IEEE 16th International Conference on Mobile Ad Hoc and Smart Systems, MASS 2019*, pp. 476–484, Nov. 2019, doi: 10.1109/MASS.2019.00062.

[3] "System brings deep learning to 'internet of things' devices | MIT News | Massachusetts Institute of Technology." Accessed: Dec. 15, 2023. [Online]. Available: https://news.mit.edu/2020/iot-deep-learning-1113

[4] Kamaldeep, M. Dutta, and J. Granjal, "Towards a Secure Internet of Things: A Comprehensive Study of Second Line Defense Mechanisms," *IEEE Access*, vol. 8, pp. 127272–127312, 2020, doi: 10.1109/ACCESS.2020.3005643.

[5] A. O. Alzahrani and M. J. F. Alenazi, "future internet Designing a Network Intrusion Detection System Based on Machine Learning for Software Defined Networks," 2021, doi: 10.3390/fi.

[6] B. Kolukisa, V. C. Yildirim, C. Ayyildiz, and V. C. Gungor, "A deep neural network approach with hyper-parameter optimization for vehicle type classification using 3-D magnetic sensor," *Comput Stand Interfaces*, vol. 84, 2023, doi: 10.1016/j.csi.2022.103703.

[7] B. Kolukisa, V. C. Yildirim, B. Elmas, C. Ayyildiz, and V. C. Gungor, "Deep learning approaches for vehicle type classification with 3-D magnetic sensor," *Computer Networks*, vol. 217, 2022, doi: 10.1016/j.comnet.2022.109326.

[8] B. Kolukisa, B. Kagan Dedeturk, H. Hacilar, and V. C. Gungor, "An efficient network intrusion detection approach based on logistic regression model and parallel artificial bee colony algorithm," *Comput Stand Interfaces*, vol. 89, p. 103808, 2024, doi: 10.1016/j.csi.2023.103808.

[9] "www.oica.net." Accessed: Dec. 06, 2023. [Online]. Available: https://www.oica.net/

[10] F. H. Somda, H. Cormerais, and J. Buisson, "Intelligent transportation systems: A safe, robust and comfortable strategy for longitudinal monitoring," *IET Intelligent Transport Systems*, vol. 3, no. 2, 2009, doi: 10.1049/iet-its:20080042.

[11] W. Ma *et al.*, "A wireless accelerometer-based automatic vehicle classification prototype system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 1, 2014, doi: 10.1109/TITS.2013.2273488.

[12] D. Obertov and B. Andrievsky, "Vehicle classification using measurements from accelerometers mounted on the road surface," in *2014 19th International Conference on Methods and Models in Automation and Robotics, MMAR 2014*, 2014. doi: 10.1109/MMAR.2014.6957389.

[13] J. George, L. Mary, and K. S. Riyas, "Vehicle detection and classification from acoustic signal using ANN and KNN," in *2013 International Conference on Control Communication and Computing, ICCC 2013*, 2013. doi: 10.1109/ICCC.2013.6731694.

[14] H. Liu, J. Ma, T. Xu, W. Yan, L. Ma, and X. Zhang, "Vehicle Detection and Classification Using Distributed Fiber Optic Acoustic Sensing," *IEEE Trans Veh Technol*, vol. 69, no. 2, 2020, doi: 10.1109/TVT.2019.2962334.

[15] S. Meta and M. G. Cinsdikici, "Vehicle-classification algorithm based on component analysis for single-loop inductive detector," *IEEE Trans Veh Technol*, vol. 59, no. 6, 2010, doi: 10.1109/TVT.2010.2049756.
[16] M. Wasilewska and B. Golenko, "Convolutional neural network based vehicle classification," in *2019 Signal Processing Symposium, SPSympo 2019*, 2019. doi: 10.1109/SPS.2019.8882050.
[17] M. I. Asborno, C. G. Burris, and S. Hernandez, "Truck Body-Type Classification using Single-Beam Lidar Sensors," *Transp Res Rec*, vol. 2673, no. 1, 2019, doi: 10.1177/0361198118821847.
[18] S. A. Rajab, A. Mayeli, and H. H. Refai, "Vehicle classification and accurate speed calculation using multi-element piezoelectric sensor," in *IEEE Intelligent Vehicles Symposium, Proceedings*, 2014. doi: 10.1109/IVS.2014.6856432.
[19] M. Stocker, M. Ronkko, and M. Kolehmainen, "Situational knowledge representation for traffic observed by a pavement vibration sensor network," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 4, 2014, doi: 10.1109/TITS.2013.2296697.
[20] H. Zhao, D. Wu, M. Zeng, and S. Zhong, "A Vibration-Based Vehicle Classification System using Distributed Optical Sensing Technology," *Transp Res Rec*, vol. 2672, no. 43, 2018, doi: 10.1177/0361198118775840.
[21] M. Bottero, B. Dalla Chiara, and F. P. Deflorio, "Wireless sensor networks for traffic monitoring in a logistic centre," *Transp Res Part C Emerg Technol*, vol. 26, 2013, doi: 10.1016/j.trc.2012.06.008.
[22] S. Taghvaeeyan and R. Rajamani, "Portable roadside sensors for vehicle counting, classification, and speed measurement," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 1, 2014, doi: 10.1109/TITS.2013.2273876.
[23] B. Yang and Y. Lei, "Vehicle detection and classification for low-speed congested traffic with anisotropic magnetoresistive sensor," *IEEE Sens J*, vol. 15, no. 2, 2015, doi: 10.1109/JSEN.2014.2359014.
[24] F. Li and Z. Lv, "Reliable vehicle type recognition based on information fusion in multiple sensor networks," *Computer Networks*, vol. 117, 2017, doi: 10.1016/j.comnet.2017.02.013.
[25] W. Balid, H. Tafish, and H. H. Refai, "Intelligent Vehicle Counting and Classification Sensor for Real-Time Traffic Surveillance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 6, 2018, doi: 10.1109/TITS.2017.2741507.
[26] H. Dong, X. Wang, C. Zhang, R. He, L. Jia, and Y. Qin, "Improved Robust Vehicle Detection and Identification Based on Single Magnetic Sensor," *IEEE Access*, vol. 6, 2018, doi: 10.1109/ACCESS.2018.2791446.
[27] C. Xu, Y. Wang, X. Bao, and F. Li, "Vehicle classification using an imbalanced dataset based on a single magnetic sensor," *Sensors (Switzerland)*, vol. 18, no. 6, 2018, doi: 10.3390/s18061690.
[28] X. Zhang and H. Huang, "Vehicle Classification Based on Feature Selection with Anisotropic Magnetoresistive Sensor," *IEEE Sens J*, vol. 19, no. 21, 2019, doi: 10.1109/JSEN.2019.2928828.
[29] X. Chen, X. Kong, M. Xu, K. Sandrasegaran, and J. Zheng, "Road Vehicle Detection and Classification Using Magnetic Field Measurement," *IEEE Access*, vol. 7, 2019, doi: 10.1109/ACCESS.2019.2908006.
[30] W. Li, Z. Liu, Y. Hui, L. Yang, R. Chen, and X. Xiao, "Vehicle Classification and Speed Estimation Based on a Single Magnetic Sensor," *IEEE Access*, vol. 8, 2020, doi: 10.1109/ACCESS.2020.3008483.

[31] J. Chang, L. Wang, G. Meng, S. Xiang, and C. Pan, "Vision-based occlusion handling and vehicle classification for traffic surveillance systems," *IEEE Intelligent Transportation Systems Magazine*, vol. 10, no. 2, 2018, doi: 10.1109/MITS.2018.2806619.

[32] E. H. Ng, S. L. Su-Lim Tan, and J. G. Guzmant, "Road traffic monitoring using a wireless vehicle sensor network," in *2008 International Symposium on Intelligent Signal Processing and Communication Systems, ISPACS 2008*, 2009. doi: 10.1109/ISPACS.2009.4806673.

[33] R. Espinosa, D. García-Saiz, M. Zorrilla, J. J. Zubcoff, and J. N. Mazón, "S3Mining: A model-driven engineering approach for supporting novice data miners in selecting suitable classifiers," *Comput Stand Interfaces*, vol. 65, 2019, doi: 10.1016/j.csi.2019.03.004.

[34] Y. Song, "Web service reliability prediction based on machine learning," *Comput Stand Interfaces*, vol. 73, 2021, doi: 10.1016/j.csi.2020.103466.

[35] S. H. Chin, C. Lu, P. T. Ho, Y. F. Shiao, and T. J. Wu, "Commodity anti-counterfeiting decision in e-commerce trade based on machine learning and Internet of Things," *Comput Stand Interfaces*, vol. 76, 2021, doi: 10.1016/j.csi.2020.103504.

[36] A. A. Afuwape, Y. Xu, J. H. Anajemba, and G. Srivastava, "Performance evaluation of secured network traffic classification using a machine learning approach," *Comput Stand Interfaces*, vol. 78, 2021, doi: 10.1016/j.csi.2021.103545.

[37] P. K. Roy, A. K. Tripathy, T. H. Weng, and K. C. Li, "Securing social platform from misinformation using deep learning," *Comput Stand Interfaces*, vol. 84, 2023, doi: 10.1016/j.csi.2022.103674.

[38] "CC1312R data sheet, product information and support | TI.com." Accessed: Dec. 06, 2023. [Online]. Available: https://www.ti.com/product/CC1312R

[39] "Skyworks | Home." Accessed: Dec. 06, 2023. [Online]. Available: https://www.skyworksinc.com/

[40] "Teach, learn, and make with the Raspberry Pi Foundation." Accessed: Dec. 06, 2023. [Online]. Available: https://www.raspberrypi.org/

[41] L. Breiman, "Random forests," *Mach Learn*, vol. 45, no. 1, pp. 5–32, Oct. 2001, doi: 10.1023/A:1010933404324/METRICS.

[42] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016. doi: 10.1145/2939672.2939785.

[43] D. W. Hosmer, S. Lemeshow, and R. X. Sturdivant, *Applied Logistic Regression: Third Edition*. 2013. doi: 10.1002/9781118548387.

[44] O. Oyebode and D. E. Ighravwe, "Urban water demand forecasting: A comparative evaluation of conventional and soft computing techniques," *Resources*, vol. 8, no. 3, 2019, doi: 10.3390/RESOURCES8030156.

[45] E. R. Girden, *ANOVA : Repeated Measures Sage University Papers Series. Quantitative Applications in the Social Sciences ; No. 07-084*. 1992.

[46] M. Kuhn and K. Johnson, *Feature engineering and selection: A practical approach for predictive models*. 2019. doi: 10.1201/9781315108230.

[47] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal Loss for Dense Object Detection," *IEEE Trans Pattern Anal Mach Intell*, vol. 42, no. 2, 2020, doi: 10.1109/TPAMI.2018.2858826.

[48] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, “Focal Loss for Dense Object Detection,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017. doi: 10.1109/ICCV.2017.324.
[49] T. Yu and H. Zhu, “Hyper-Parameter Optimization: A Review of Algorithms and Applications,” Mar. 2020, [Online]. Available: http://arxiv.org/abs/2003.05689
[50] L. Yao, Z. Fang, Y. Xiao, J. Hou, and Z. Fu, “An Intelligent Fault Diagnosis Method for Lithium Battery Systems Based on Grid Search Support Vector Machine,” *Energy*, vol. 214, 2021, doi: 10.1016/j.energy.2020.118866.
[51] “Hyperparameter tuning. Grid search and random search | Your Data Teacher.” Accessed: Jan. 11, 2024. [Online]. Available: https://www.yourdatateacher.com/2021/05/19/hyperparameter-tuning-grid-search-and-random-search/
[52] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual, Scotts Valley*. 2009.
[53] F. Pedregosa *et al.*, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, 2011.
[54] “GitHub - keras-team/keras: Deep Learning for humans.” Accessed: Dec. 06, 2023. [Online]. Available: https://github.com/keras-team/keras
[55] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: Synthetic minority over-sampling technique,” *Journal of Artificial Intelligence Research*, vol. 16, 2002, doi: 10.1613/jair.953.
[56] S. J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10. 2010. doi: 10.1109/TKDE.2009.191.
[57] R. M. Schmidt, “Recurrent Neural Networks ( RNNs ): A gentle Introduction and Overview arXiv : 1912 . 05911v1 [ cs . LG ] 23 Nov 2019,” *ArXiv*, no. 1, 2019.
[58] “A Brief Introduction to Recurrent Neural Networks | by Jonte Dancker | Towards Data Science.” Accessed: Jan. 11, 2024. [Online]. Available: https://towardsdatascience.com/a-brief-introduction-to-recurrent-neural-networks-638f64a61ff4
[59] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink, and J. Schmidhuber, “LSTM: A Search Space Odyssey,” *IEEE Trans Neural Netw Learn Syst*, vol. 28, no. 10, 2017, doi: 10.1109/TNNLS.2016.2582924.
[60] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Comput*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/NECO.1997.9.8.1735.
[61] K. Cho *et al.*, “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation,” Jun. 2014, [Online]. Available: http://arxiv.org/abs/1406.1078
[62] E. Baykal, H. Dogan, M. E. Ercin, S. Ersoz, and M. Ekinci, “Transfer learning with pre-trained deep convolutional neural networks for serous cell classification,” *Multimed Tools Appl*, vol. 79, no. 21–22, 2020, doi: 10.1007/s11042-019-07821-9.
[63] B. Kolukısa, Y. Görmez, and Z. Aydın, “A Transfer Learning Approach for Skin Cancer Subtype Detection,” 2023. doi: 10.1007/978-3-031-31956-3_28.
[64] “The Unreasonable Effectiveness of Recurrent Neural Networks.” Accessed: Jan. 11, 2024. [Online]. Available: https://karpathy.github.io/2015/05/21/rnn-effectiveness/
[65] “Image data loading.” Accessed: Dec. 06, 2023. [Online]. Available: https://keras.io/api/data_loading/image/

[66] "Digital 2019: Global Digital Overview — DataReportal – Global Digital Insights." Accessed: Dec. 06, 2023. [Online]. Available: https://datareportal.com/reports/digital-2019-global-digital-overview
[67] SonicWall, "2021 SonicWall Cyber Threat Report," 2021.
[68] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *Proceedings - IEEE Symposium on Security and Privacy*, 2010. doi: 10.1109/SP.2010.25.
[69] A. Thakkar and R. Lohiya, "Role of swarm and evolutionary algorithms for intrusion detection system: A survey," *Swarm Evol Comput*, vol. 53, 2020, doi: 10.1016/j.swevo.2019.100631.
[70] Bäck, T., Fogel, D. B., & Michalewicz, Z. (Eds.). (2018). Evolutionary computation 1: Basic algorithms and operators. CRC press.
[71] V. R. Balasaraswathi, M. Sugumaran, and Y. Hamid, "Feature selection techniques for intrusion detection using non-bio-inspired and bio-inspired optimization algorithms," *Journal of Communications and Information Networks*, vol. 2, no. 4, 2017, doi: 10.1007/s41650-017-0033-7.
[72] Peltier, T. R. (2016). Information Security Policies, Procedures, and Standards: guidelines for effective information security management. CRC press.
[73] Y. Han, M. Yang, H. Qi, X. He, and S. Li, "The improved logistic regression models for spam filtering," in *2009 International Conference on Asian Language Processing: Recent Advances in Asian Language Processing, IALP 2009*, 2009. doi: 10.1109/IALP.2009.74.
[74] D. Karaboga, "An idea based on Honey Bee Swarm for Numerical Optimization," *Technical Report TR06, Erciyes University*, no. TR06, 2005.
[75] D. Karaboga and B. Akay, "A comparative study of Artificial Bee Colony algorithm," *Appl Math Comput*, vol. 214, no. 1, 2009, doi: 10.1016/j.amc.2009.03.090.
[76] B. Akay and D. Karaboga, "A modified Artificial Bee Colony algorithm for real-parameter optimization," *Inf Sci (N Y)*, vol. 192, 2012, doi: 10.1016/j.ins.2010.07.015.
[77] H. Liu and B. Lang, "Machine learning and deep learning methods for intrusion detection systems: A survey," *Applied Sciences (Switzerland)*, vol. 9, no. 20. 2019. doi: 10.3390/app9204396.
[78] V. Hajisalem and S. Babaie, "A hybrid intrusion detection system based on ABC-AFS algorithm for misuse and anomaly detection," *Computer Networks*, vol. 136, 2018, doi: 10.1016/j.comnet.2018.02.028.
[79] A. U. H. Qureshi, H. Larijani, N. Mtetwa, A. Javed, and J. Ahmad, "RNN-ABC: A new swarm optimization based technique for anomaly detection," *Computers*, vol. 8, no. 3, 2019, doi: 10.3390/computers8030059.
[80] M. Mazini, B. Shirazi, and I. Mahdavi, "Anomaly network-based intrusion detection system using a reliable hybrid artificial bee colony and AdaBoost algorithms," *Journal of King Saud University - Computer and Information Sciences*, vol. 31, no. 4, 2019, doi: 10.1016/j.jksuci.2018.03.011.
[81] T. Gu, H. Chen, L. Chang, and L. Li, "Intrusion detection system based on improved abc algorithm with tabu search," *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 14, no. 11, 2019, doi: 10.1002/tee.22987.
[82] "Employing Artificial Bee Colony Algorithm for Feature Selection in Intrusion Detection System | IEEE Conference Publication | IEEE Xplore." Accessed: Dec. 22, 2023. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9441363

[83] M. Rani and Gagandeep, "Effective network intrusion detection by addressing class imbalance with deep neural networks multimedia tools and applications," *Multimed Tools Appl*, vol. 81, no. 6, 2022, doi: 10.1007/s11042-021-11747-6.
[84] B. K. Dedeturk and B. Akay, "Spam filtering using a logistic regression model trained by an artificial bee colony algorithm," *Applied Soft Computing Journal*, vol. 91, 2020, doi: 10.1016/j.asoc.2020.106229.
[85] S. M. Kasongo and Y. Sun, "Performance Analysis of Intrusion Detection Systems Using a Feature Selection Method on the UNSW-NB15 Dataset," *J Big Data*, vol. 7, no. 1, 2020, doi: 10.1186/s40537-020-00379-6.
[86] S. Solani and N. K. Jadav, "A novel approach to reduce false-negative alarm rate in network-based intrusion detection system using linear discriminant analysis," in *Lecture Notes in Networks and Systems*, 2021. doi: 10.1007/978-981-15-7345-3_77.
[87] S. Meftah, T. Rachidi, and N. Assem, "Network based intrusion detection using the UNSW-NB15 dataset," *International Journal of Computing and Digital Systems*, vol. 8, no. 5, 2019, doi: 10.12785/ijcds/080505.
[88] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Shallow neural network with kernel approximation for prediction problems in highly demanding data networks," *Expert Syst Appl*, vol. 124, 2019, doi: 10.1016/j.eswa.2019.01.063.
[89] D. Jing and H. B. Chen, "SVM based network intrusion detection for the UNSW-NB15 dataset," in *Proceedings of International Conference on ASIC*, 2019. doi: 10.1109/ASICON47005.2019.8983598.
[90] N. Moustafa and 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 2015 - Proceedings*, 2015. doi: 10.1109/MilCIS.2015.7348942.
[91] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *IEEE Symposium on Computational Intelligence for Security and Defense Applications, CISDA 2009*, 2009. doi: 10.1109/CISDA.2009.5356528.
[92] C. Huang, Y. Li, and X. Yao, "A Survey of Automatic Parameter Tuning Methods for Metaheuristics," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 2. 2020. doi: 10.1109/TEVC.2019.2921598.
[93] Y. Gormez, Z. Aydin, R. Karademir, and V. C. Gungor, "A deep learning approach with Bayesian optimization and ensemble classifiers for detecting denial of service attacks," *International Journal of Communication Systems*, vol. 33, no. 11, 2020, doi: 10.1002/dac.4401.
[94] C. R. Harris *et al.*, "Array programming with NumPy," *Nature*, vol. 585, no. 7825. 2020. doi: 10.1038/s41586-020-2649-2.
[95] R. Okuta, Y. Unno, D. Nishino, S. Hido, and C. Loomis, "CuPy: A NumPy-Compatible Library for NVIDIA GPU Calculations." [Online]. Available: https://github.com/cupy/cupy.
[96] "abcLR · PyPI." Accessed: Dec. 06, 2023. [Online]. Available: https://pypi.org/project/abcLR/
[97] "GitHub - kagandedeturk/ABC-LR." Accessed: Dec. 06, 2023. [Online]. Available: https://github.com/kagandedeturk/ABC-LR

# CURRICULUM VITAE

| | |
|---|---|
| 2011 – 2016 | B.Sc., Computer Engineering,<br>Erciyes University, Kayseri, TÜRKİYE |
| 2018 – 2020 | M.Sc., Electrical and Computer Engineering,<br>Abdullah Gül University, Kayseri, TÜRKİYE |
| 2020 – 2024 | Ph.D., Electrical and Computer Engineering,<br>Abdullah Gül University, Kayseri, TÜRKİYE |
| 2018 – present | Research Assistant, Computer Engineering,<br>Abdullah Gül University, Kayseri, TÜRKİYE |

SELECTED PUBLICATIONS AND PRESENTATIONS

**J1)** B. Kolukisa, B. K. Dedeturk, H. Hacilar, V. C. Gungor, An efficient network intrusion detection approach based on logistic regression model and parallel artificial bee colony algorithm, published in Elsevier Journal of Computer Standards & Interfaces (2024).

**J2)** B. Kolukisa, V. C. Yildirim, C. Ayyildiz, V. C. Gungor, A deep neural network approach with hyper-parameter optimization for vehicle type classification using 3-D magnetic sensor, published in Elsevier Journal of Computer Standards & Interfaces (2023).

**J3)** B. Kolukisa, B. Bakir-Gungor, Ensemble feature selection and classification methods for machine learning-based coronary artery disease diagnosis, published in Elsevier Journal of Computer Standards & Interfaces (2023).

**J4)** B. Adanur-Dedeturk, B. Kolukisa, S. Tonyali, Privacy-Preserving Wireless Indoor Localization Systems, published in DergiPark Kocaeli Journal of Science and Engineering (2023).

**J5)** B. K. Dedeturk, B. Kolukisa, M. Ozmen, CLUSTER-BASED CLONAL SELECTION ALGORITHM FOR VEHICLE ROUTING PROBLEMS WITH TIME WINDOWS, published in DergiPark Journal of Adıyaman Üniversitesi Mühendislik Bilimleri Dergisi (2023).

**J6)** B. Kolukisa, V. C. Yildirim, B. Elmas, C. Ayyildiz, V. C. Gungor, Deep learning approaches for vehicle type classification with 3-D magnetic sensor, published in Elsevier Journal of Computer Networks (2022).

**J7)** B. Kolukisa, L. Yavuz, A. Soran, B. Bakir-Gungor, D. Tuncer, A. Onen, V. C. Gungor, Coronary artery disease diagnosis using optimized adaptive ensemble machine learning algorithm, published in International Journal of Bioscience, Biochemistry and Bioinformatics (2020).

**J8)** B. Kolukisa, H. Hacilar, M. Kuş, B. Bakır-Gungor, A. Aral, V. C. Gungor, Diagnosis of coronary heart disease via classification algorithms and a new feature selection methodology, published in International Journal of Data Mining Science (2019).

**J9)** G. Goy, B. Kolukisa, B. Bakir-Gungor, I. Ugur, V. C Gungor, Weighted Association Rules and Scoring Methodology for Cardiovascular Diseases, published in International Journal of Bioscience, Biochemistry and Bioinformatics (2019).

**C1)** B. Kolukisa, Y. Gormez, Z. Aydin, A Transfer Learning Approach for Skin Cancer Subtype Detection, In International Conference on Artificial Intelligence and Applied Mathematics in Engineering Cham: Springer International Publishing (2022).

**C2)** B. Kolukisa, B. K. Dedeturk, B. A. Dedeturk, A. Gulsen, G. Bakal, A Comparative Analysis on Medical Article Classification Using Text Mining & Machine Learning Algorithms. In 6th IEEE International Conference on Computer Science and Engineering (2021).

**C3)** B. Kolukisa, V. C. Gungor, B. Bakir-Gungor, An Ensemble Feature Selection Methodology That Incorporates Domain Knowledge for Cardiovascular Disease Diagnosis. In 28th IEEE Signal Processing and Communications Applications Conference (2020)

**C4)** G. GOY, B. Kolukisa, C. Bahcevan, V. C. Gungor, Ensemble churn prediction for internet service provider with machine learning techniques, In 5th IEEE International Conference on Computer Science and Engineering (2019)

**C5)** B. Kolukisa, H. Hacilar, G. Goy, M. Kus, B. Bakir-Gungor, A. Aral, V. C. Gungor, Evaluation of classification algorithms, linear discriminant analysis and a new hybrid feature selection methodology for the diagnosis of coronary artery disease, In IEEE International Conference on Big Data (2018).