Hilal HACILAR

Ph.D. Thesis

AGU 2024

MACHINE LEARNING BASED 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
Hilal HACILAR
August 2024

MACHINE LEARNING BASED 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
Hilal HACILAR
August 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: Hilal HACILAR

Signature :

REGULATORY COMPLIANCE

Ph.D. thesis titled “Machine Learning Based Network Anomaly Detection” has been prepared in accordance with the Thesis Writing Guidelines of the Abdullah Gül University, Graduate School of Engineering & Science.

Prepared By	Advisor	Co-Advisor
Hilal HACILAR	Assoc. Prof. Burcu BAKIR GÜNGÖR	Prof. V. Çağrı GÜNGÖR
Signature	Signature	Signature

Head of the Electrical and Computer Engineering Program

Assoc. Prof. Samet GÜLER

Signature

ACCEPTANCE AND APPROVAL

Ph.D. thesis titled “Machine Learning Based Network Anomaly Detection” and prepared by Hilal HACILAR has been accepted by the jury in the the Electrical and Computer Engineering Graduate Program at Abdullah Gül University, Graduate School of Engineering & Science.

22 /08/ 2024

(Thesis Defense Exam Date)

JURY:

Advisor : Assoc. Prof. Burcu BAKIR GÜNGÖR

Co-Advisor : Prof. V. Çağrı GÜNGÖR

Member : Assoc. Prof. Özkan Ufuk NALBANTOĞLU

Member : Asst. Prof. Abdulkadir KÖSE

Member : Assoc. Prof. Rifat Kurban

Member : Asst. Prof. Tayyip Özcan

APPROVAL:

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

…….. /…….. / ……..

(Date)

Graduate School Dean
Prof. İrfan ALAN

ABSTRACT
MACHINE LEARNING BASED NETWORK ANOMALY DETECTION

Hilal HACILAR
Ph.D. in Electrical and Computer Engineering
Advisor: Assoc. Prof. Burcu BAKIR GÜNGÖR
Co-Advisor: Prof. V. Çağrı GÜNGÖR
August 2024

Intelligent technologies have led to a significant rise in internet users and applications. However, this rise in internet usage has also brought serious security challenges. Organizations rely on Network Intrusion Detection systems (NIDS) to protect sensitive data from unauthorized access and theft. To enhance the capabilities of IDS, Machine Learning (ML) and Deep Learning (DL) techniques have been increasingly integrated into these systems. In this context, anomaly-based network intrusion detection surpasses other detection mechanisms significantly in several instances. These systems analyze network traffic to detect suspicious activities, such as attempted breaches or cyberattacks. However, existing studies lack a thorough assessment of class imbalances, feature selection and extraction methods, hyperparameter optimization, and classification performance for different types of network intrusions: wired, wireless, and Software Defined Networking (SDN). Additionally, existing methods may achieve high accuracy; they may suffer from high training times, low detection rate (DR), and computational complexity. By combining metaheuristics and neural networks, it is possible to solve complex optimization problems that are challenging to solve using conventional methods. To address these challenges, this thesis study first evaluates different network intrusion datasets, such as wired, wireless, and SDN, together, considering class imbalance, feature selection, and hyperparameter optimization tasks. Secondly, it proposes a novel hybrid approach combining Deep Autoencoder (DAE) and Artificial Neural Network (ANN) models trained by a parallel Artificial Bee Colony (ABC) algorithm with Bayesian hyperparameter optimization.

Keywords: Network Intrusion Detection systems (NIDS), Network Anomaly Detection, Machine Learning (ML), Deep Learning (DL), Metaheuristics.

ÖZET

MAKİNE ÖĞRENMESİ TABANLI AĞ ANOMALİ TESPİTİ

Hilal HACILAR
Elektrik ve Bilgisayar Mühendisliği Anabilim Dalı Doktora
Tez Danışmanı: Doç. Dr. Burcu BAKIR GÜNGÖR
İkinci Tez Danışmanı: Prof. Dr. V. Çağrı GÜNGÖR
Ağustos 2024

Akıllı teknolojiler, internet kullanıcılarının ve uygulamalarının önemli ölçüde artmasına neden olmuştur. Ancak, internet kullanımındaki bu artış ciddi güvenlik sıkıntılarını da beraberinde getirmiştir. Kuruluşlar, hassas verileri yetkisiz erişim ve hırsızlıktan korumak için Ağ Saldırı Tespit Sistemlerine (NIDS) güvenmektedir. IDS'nin yeteneklerini artırmak amacıyla, makine öğrenimi (ML) ve derin öğrenme (DL) teknikleri giderek daha fazla bu sistemlere entegre edilmektedir. Bu bağlamda, anomali tabanlı ağ saldırı tespiti, birçok durumda diğer tespit mekanizmalarını önemli ölçüde geride bırakmaktadır. Bu sistemler, ağ trafiğini analiz ederek, saldırı girişimleri veya siber saldırılar gibi şüpheli faaliyetleri tespit etmektedir. Ancak, mevcut çalışmalar, kablolu, kablosuz ve Yazılım Tanımlı Ağlar (SDN) gibi farklı türde ağ saldırıları için sınıf dengesizlikleri, özellik seçimi ve çıkarma yöntemleri, hiperparametre optimizasyonu ve sınıflandırma performansı konularında kapsamlı bir değerlendirmeden yoksundur. Ayrıca, mevcut yöntemler yüksek doğruluk elde edebilirken, yüksek eğitim süreleri, düşük tespit oranları ve hesaplama karmaşıklığı gibi sorunlar yaşayabilirler. Metaheuristikler ve sinir ağlarını birleştirerek, geleneksel yöntemlerle çözülmesi zor olan karmaşık optimizasyon problemlerini çözmek mümkündür. Bu zorlukları ele almak için, bu tez çalışması ilk olarak, kablolu, kablosuz ve SDN gibi farklı ağ saldırı veri setlerini sınıf dengesizliği, özellik seçimi ve hiperparametre optimizasyonu görevlerini dikkate alarak birlikte değerlendirmektedir. İkinci olarak, Bayes hiperparametre optimizasyonu ile paralel yapay arı kolonisi algoritması tarafından eğitilen Derin Otomatik Kodlayıcı ve ANN modellerini birleştiren yeni bir hibrit yaklaşım önermektedir.

Anahtar kelimeler: Ağ Saldırı Tespit Sistemleri, Ağ Anomali Tespiti, Makine Öğrenimi, Derin Öğrenme, Metaheuristikler.

Acknowledgements

First and foremost, I wish to express my sincere appreciation to my co-advisor, Prof. Vehbi Çağrı Güngör, for his invaluable guidance and patience throughout the entirety of my Ph.D. journey. I am profoundly grateful for the depth of experience he has brought to this journey, as well as for everything he has taught me, both academically and personally. Additionally, I would like to express my gratitude to my supervisor, Assoc. Prof. Burcu Bakır Güngör, for her valuable contributions and support.

I want to extend my heartfelt thanks to my wonderful collages Zeynep Şenel and Merve Balki Taş for their unwavering support and companionship they've provided me with during this journey.

I would like to thank YÖK for providing 100/2000 YÖK Ph.D. scholarship.

I would like to give heartfelt thanks to my beloved immediate family. Azra, my first steps into academia coincided with your first steps in this world. Vera, as you came into our lives, you brought with you a renewed sense of purpose and determination. I am grateful for the beautiful emotions they have evoked in me and the spiritual support they have provided throughout this journey. I want to express my sincere appreciation to my beloved spouse, Ahmet Hacılar, for his unwavering support and patience.

Furthermore, I would like to extend my sincere gratitude to my twin sister, Elif Kurşuncu, my beloved mother, Leman Muğaloğlu, and all the members of my 'Muğaloğlu' and 'Hacılar' families for their boundless support and understanding throughout this journey.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ABC	Artificial Bee Colony
Adasyn	Adaptive Synthetic Sampling
AE	Autoencoder
ANN	Artificial Neural Network
BWO	Black Window Optimization
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DA	Dragonfly Algorithm
DAE	Deep Autoencoder
DBN	Deep Belief Network
DDos	Distributed Denial of Service
DL	Deep Learning
DNN	Deep Neural Network
DR	Detection Rate
DT	Decision Tree
FAR	False Alarm Rate
GA	Genetic Algorithm
GD	Gradient Descent
GDN	Graph Deviation Network
GNN	Graph Neural Network
GPU	Graphics Processing Unit
IoT	Internet of Things
kNN	K Nearest Neighbors
LDA	Linear Discriminant Analysis
LM	Levenberg-Marquardt

LSTM	Long Short-Term Memory
ML	Machine Learning
MLP	Multi Layer Perceptron
NB	Naive Bayes
NIDS	Network Intrusion Detection System
PCA	Principal Component Analysis
PSO	Particle Swarm Optimization
RNN	Recurrent Neural Network
ROS	Random Over Sampling
SDN	Software-Defined networking
SIDS	Signature-based Network Intrusion Detection System
SMOTE	Synthetic Minority Over-Sampling
SMOTETomek	Synthetic Minority Over-Sampling with Tomek Links
SQP	Sequential Quadratic Programming
SVM	Support Vector Machine
TPE	Tree-structured Parzen Estimator
TPR	True Positive Rate
TTL	Time to Live
XGBoost	Extreme Gradient Boosting

To my family

Chapter 1

Introduction

The exponential growth of computer networks has led to a significant rise in internet users, applications, and security concerns, particularly regarding network intrusions. To address these concerns and vulnerabilities, various security measures like firewalls, data encryption, and user authentication are implemented. Despite their effectiveness, they lack comprehensive packet analysis, making it challenging to detect all types of attacks. To address these concerns, the detection of network attacks has become a top priority and a significant challenge for researchers in computer science and network security. Over time, ML models have increasingly been incorporated into a variety of problems to accurately detect intrusions. The identification of normal and abnormal flows and their attack types in the network is an important issue. As a solution, NIDS has been developed. NIDS continuously monitors networks for malicious activities and promptly alerts users to any intrusions or attacks, serving as an important line of defense against network threats.

Depending on the type of analysis conducted, NIDS is classified as either signature-based (SIDS) or anomaly-based (NIDS). Signature-based approaches focus on predefined patterns within the analyzed data and search for existing signatures. Hence, they establish a pre-existing signature database containing known attacks. On the other hand, anomaly-based detectors attempt to predict the system's 'normal' or 'abnormal' behavior, i.e., they issue an anomaly alert when the difference between a specific observation at a given moment and the typical behavior exceeds a predetermined threshold. Therefore, anomaly-based intrusion detection significantly outperforms other detection mechanisms in various instances, laying the foundation for this thesis study.

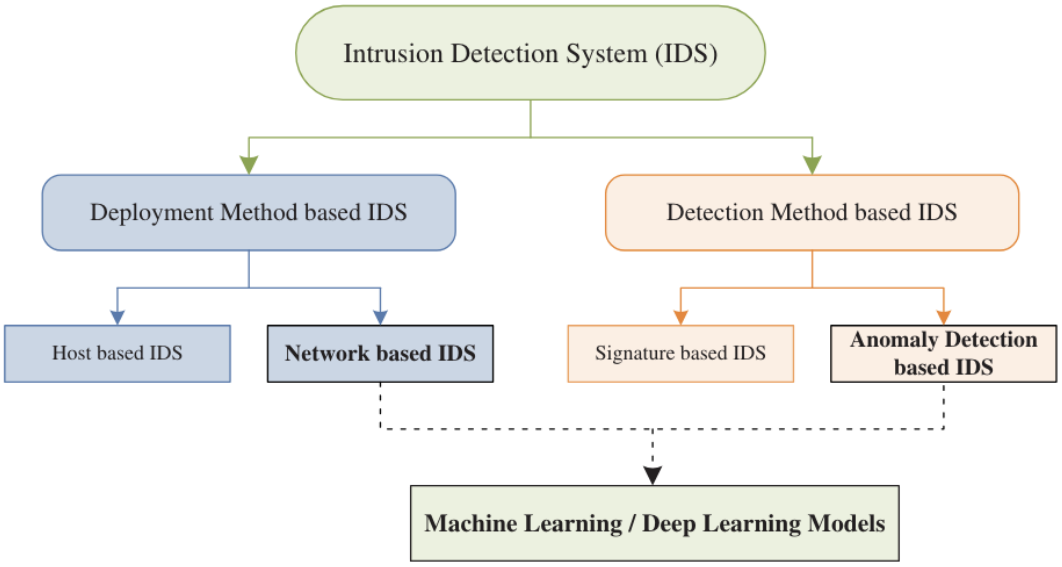

Figure 1. 1 Intrusion detection system classification taxonomy [1]

1.1 Research Problem Definition

Researchers have conducted studies on various techniques for selecting features and integrating classifiers with NIDS in order to improve the ability to predict network attacks. Nevertheless, the lack of a widely accepted approach for detecting network intrusions is obvious, and a noticeable lack of scientific studies comparing the different feature selection and classification algorithms performances with respect to distinct evaluation metrics is apparent.

The research problem encompasses the aim to develop an innovative network intrusion detection system that outperforms existing systems in terms of DR, false alarm rate (FAR), and response time, given the advancements in ML and DL.

The model should:

1. Operate at near real-time speeds.
2. Achieve high accuracy and F-measure.
3. Attain a high DR while maintaining a low FAR.

In order to address the existing research gap, the primary aim of this thesis is to conduct a comprehensive analysis of network intrusions inside wired, wireless, and SDN settings. **The first chapter** specifically focused on the challenges associated with classification and class imbalance.

While existing methods may achieve high accuracy, they may suffer from high training times, low DR, and computational complexity. In this context, metaheuristic

algorithms provide robust optimization abilities for large datasets, empowering researchers to address complex optimization challenges with efficiency and effectiveness.

ANN is a type of ML algorithm inspired by biological neural networks that mimic the learning and processing abilities of humans. These are highly effective at modeling non-linear relationships. However, designing an appropriate network structure and finding ideal weight values pose significant optimization challenges. The conventional approach for training ANNs involves weight adjustment via error back-propagation using gradient descent (GD)-based algorithms. Due to the dependence of error surfaces on parameters and initial weights, these algorithms frequently get stuck in local minima.

Numerous studies in literature have demonstrated successful outcomes by integrating various metaheuristic approaches with ANNs. However, a prevalent issue with the ABC algorithm is its extended training time, leading many studies in the literature to focus on small datasets. Those employing the ABCANN algorithm with large datasets often lack detailed information regarding training time. To address this challenge, **the second chapter** proposes novel approaches: a vectorized and parallelized DAE-based hybrid ABC-ANN algorithm for binary classification tasks and a vectorized and parallelized ABC-ANN algorithm for multi-class classification tasks. This methodology leverages the respective strengths of DAE, the ABC algorithm, and parallel computing techniques to expedite the training process. In this respect, this study suggests that the ABC algorithm, with some modifications, can avoid local minimum solutions by conducting a high-performance search in the solution space.

1.2 Contributions

Based on the studies conducted in my thesis, we can express the contributions I have made under two subsections.

1.2.1 Network Intrusion Detection based on Machine Learning strategies: Performance comparisons on imbalanced Wired, Wireless, and Software-Defined Networking (SDN) network traffics[2]

This analysis will specifically focus on the challenges associated with classification and class imbalance. This study utilizes many techniques such as class

imbalance strategies, feature selection, hyperparameter optimization, and classification methods in order to accurately detect network intrusions. In this context, this study employs various methods, including Random Over Sampling (ROS), Adaptive Synthetic Sampling (Adasyn) [3], Synthetic Minority Over-Sampling Technique (SMOTE), and SMOTETomek, to address the issue of unbalanced datasets. Additionally, Extreme Gradient Boosting **(**XGBoost) has been utilized to assess the feature importances during the feature selection process. The main contributions of this study to the literature are listed below:

1. To alleviate the computational workload and minimize training time, XGBoost feature selection has been applied to pinpoint the most informative features within IDS datasets.
2. An efficient hyperparameter optimization technique called Bayesian optimization has been employed to fine-tune the model's parameters. This optimization process aims to identify the best combination of hyperparameters that yield optimal performance while minimizing the required training time.
3. By applying imbalanced strategies including ROS, SMOTE, SMOTETomek, and Adasyn, the model can better handle the challenges posed by imbalanced datasets and improve its performance in detecting intrusions accurately while minimizing the impact of class imbalance.
4. Various ML models have been constructed to classify intrusions and normal flows, followed by a comprehensive evaluation using various metrics such as F1-measure, overall accuracy, FAR, and DR (Figure 2.1 provides an overview of these steps). It has been observed that the proposed classification methodologies produce superior results compared to the existing literature (Table 2.1).
5. As far as we are aware, no study has been conducted to evaluate different network intrusion datasets, such as wired, wireless, and SDN, together, considering class imbalance, feature selection, and hyperparameter optimization tasks. This study aims to fill this gap.

1.2.2 Network anomaly detection using deep autoencoder and parallel artificial bee colony algorithm-trained neural network

This analysis will specifically focus on the challenges with conventional ANNs and their prolonged training times, limiting their application to small datasets and

frequent entrapment in local minima. To address these issues, in this study, our main contributions are as follows:

1. As the sizes of storable and actively usable data continue to increase every day, the importance of Graphics Processing Unit **(**GPU) parallelization cannot be overlooked. Despite the ABC algorithm being a highly successful metaheuristic algorithm, there are deficiencies and gaps in the literature regarding its applicability to large datasets. This study develops an ABC-based ANN model that is simplified to work efficiently on modern hardware, reducing computational complexity as much as possible (vectorization) and making it suitable for GPU execution (parallelization). The model is tailored for both binary and multi-class classification tasks.
2. This study introduces a novel approach by proposing a DAE-based, vectorized, and parallelized ABC-ANN algorithm for binary classification task. The aim is to enhance classification accuracy and DR. By employing the DAE, the algorithm extracts relevant and distinctive features from the input data, leading to a more effective detection process.
3. An XGBoost-based feature selection approach has been implemented to reduce significant computational costs associated with typical ANN based models using ABC algorithms. This technique effectively decreases the number of dimensions in the input data, therefore reducing the computing cost.
4. Different evaluation metrics, such as accuracy, f1-measure, DR, FAR, and training time for binary classification, as well as recall-macro, recall-weighted, f1-macro (calculated as the arithmetic mean of each per-class f1 score), f1-weighted (computed as the mean of all per-class f1 scores, taking into account each class's support), precision-macro, precision-weighted, accuracy, and training time for multi-class classification , were used to test and compare how well the proposed method works with existing ML techniques. This detailed assessment offers valuable insights regarding the efficiency of the suggested approach.
5. To automatically optimize the hyperparameters of the proposed DAE-ABC-ANN approach and the metaheuristics, the Bayesian parameter optimization method is utilized. This optimization method intelligently explores the hyperparameter space, facilitating the selection of the best hyperparameter configurations for each model.

6. By incorporating these advancements, the proposed approach outperforms some metaheuristics in terms of evaluation metrics and training time.

1.3 Research Outline

The thesis has been carefully organized into chapters and sections to present the solutions to the problem in a coherent and comprehensive manner.

Chapter 2 conducts an evaluation of various network environment intrusions, including wired, wireless, and SDN, collectively considering class imbalance, feature selection, and hyperparameter optimization tasks. The organization of Chapter 2 can be sorted as follows: In the first section, we address the gaps in the literature and our motivation for conducting this study. The subsequent section examines related work on network intrusion detection. The materials and methods section introduces the datasets, providing detailed information about their network features and preprocessing steps. It then describes the class imbalance problem, approaches for feature selection and extraction using XGBoost and Autoencoders (AEs), and details the Bayesian hyperparameter optimization algorithm. The fourth section presents the evaluation metrics used in the classification conducted on the UNSW-NB15, AWID, and InSDN datasets. The final section provides experimental setups and results.

Chapter 3 proposes a novel DAE-based hybrid ABC-ANN algorithm that incorporates vectorization and GPU parallelization techniques into the ABC algorithm for the first time in the literature. These methods aim to enhance the algorithm's efficiency and exploit the capabilities of modern hardware, providing a substantial advancement in the network intrusion detection field. The organization of Chapter 2 can be sorted as follows: Related work provides an overview of literature, encompassing topics such as ABC-ANN, NIDS, and metaheuristics on NIDS. The proposed method section offers a detailed explanation of the proposed DAE-based parallel ABC-ANN method. Lastly, experimental results on two datasets, including the UNSW-NB15 and NF-UNSW-NB15-v2, along with discussions, are presented in the experimental results section.

Finally, Chapter 4 encapsulates the research findings, elaborates on the societal impact and contributions to global sustainability, and identifies several promising areas for future research.

Chapter 2

Network Intrusion Detection based on Machine Learning strategies: Performance comparisons on imbalanced Wired, Wireless, and Software-Defined Networking (SDN) network traffics

2.1 Motivation

Efficient and accurate intrusion detection is of great importance to safeguarding network security against evolving threats. However, traditional approaches often deal with computational complexity and prolonged training times, limiting their efficacy for timely threat detection. To address these challenges, this study aims to innovate and optimize intrusion detection methodologies.

The primary objective of NIDS is to develop predictive models capable of distinguishing normal network activities from abnormal ones. Various studies in the literature have explored predictive modeling techniques, including XGBoost, ANN, DL, and other conventional ML algorithms. However, class imbalance problems often arise in these predictive modeling studies [4]. Many approaches, like SMOTE [5], have been suggested as potential solutions to tackle this issue and have shown their efficacy by improving classification accuracy. In conjunction with the issue of class imbalance, an effective network anomaly detection system must also address the challenges posed by

various attack patterns, encrypted data transmissions, and the need for real-time application performance.

Given these challenges, the motivation behind our work is to create an NIDS model that can effectively operate in the wired, wireless, and SDN network domains while considering aspects such as class imbalance, feature selection, and hyperparameter optimization.

2.2 Related Work

The literature on network anomaly detection suggests a wide amount of algorithms, including both DL techniques and standard ML algorithms. Researchers have emphasized the importance of rapid response in network intrusion detection systems. For instance, Potluri et al. [6] have utilized parallel computing and DL algorithms on the NSL-KDD dataset, achieving an accuracy score of 97.5%. Hoang et al. [7] have employed parallel genetic programming on the AWID dataset, reducing computing costs and achieving precision, recall, and F1-measure values of 0.785, 0.78, and 0.78, respectively. Kolukisa et al. [8] also trained logistic regression parameters via the parallel ABC algorithm and achieved significant results on the UNSW_NB15 dataset.

Tree-based ML algorithms, such as XGBoost, have demonstrated effectiveness and feasibility in NIDS [9] [10] [11] [12]. Kevric et al. [9] utilized NB trees and random trees as a hybrid technique on the NSL-KDD dataset, achieving an accuracy of 89.24%. Pattawaro et al. [10] employed an ensemble approach combining K Nearest Neighbors (kNN) and XGBoost on the NSL-KDD dataset, achieving an accuracy of 84.4%. Dhaliwal et al. [12] performed XGBoost on the NSL-KDD dataset, attaining values of 98.70% accuracy and 98.76% F1-score. In another study [13], a novel hybrid feature selection and LightGBM-based Intrusion Detection System (IDS) is proposed for SDN. They tested their proposed model on the NSL-KDD dataset and achieved notable results.

Prior to classification, several studies have employed dimension reduction techniques such as AE, Linear Discriminant Analysis (LDA), and Principal Component Analysis (PCA). LDA and PCA are linear transformation methods, with PCA being unsupervised and LDA being supervised. AEs, on the other hand, are neural networks that attempt to compress input data into a smaller model using an encoder and a

bottleneck. AEs operate in a nonlinear manner, similar to PCA but with greater flexibility. Research studies have generated AE algorithms for dimension reduction and classification tasks, achieving high evaluation scores [14], [15], [16], [17], [18], [19].

Wheelus et al. [20] tackle the issue mentioned and demonstrate that SMOTE proves to be more effective than alternative algorithms in addressing class imbalance, particularly in terms of ROC area compared to specific ML algorithms. Additionally, Abdulhammed et al. [21] acknowledge the challenge of class imbalance and implement preprocessing techniques designed for imbalanced datasets, achieving a remarkable 99.99% accuracy in the CIDDS-001 intrusion dataset. Meanwhile, Ran et al. [22] adopt a semi-supervised learning approach as a substitute for traditional supervised ML algorithms, incorporating undersampling to effectively address the class imbalance problem. Abdelkhalek and Mashaly [23] achieve significant results by combining the Adasyn and TomekLinks sampling techniques.

Other DL methods, including Deep Belief Networks (DBN), Deep Neural Networks (DNN), Long Short-Term Memory (LSTM)-based AE, Convolutional Neural Networks (CNN), and Recurrent Neural Networks (RNN), have been utilized to construct predictive models. These approaches have demonstrated high accuracy rates on datasets such as NSL-KDD, KDD Cup '99, and InSDN [19], [20], [24], [25]. Some studies achieve significant results by employing DL algorithms in a hybrid manner. For instance, Qazi et al. [26] apply CNN and RNN algorithms in a hybrid manner to the CICIDS-2018 dataset, resulting in notable outcomes.

Regarding SDN, research studies have highlighted the importance of detecting Distributed Denial of Service (DDoS) attacks in SDN networks. Bhayo et al. [27] proposed ML-based approach to detect DDoS attacks in an SDN-WISE Internet of Things (IoT) controller. They integrated Naive Bayes (NB), Support Vector Machine (SVM), and Decision Tree (DT)-based detection modules into the controller and achieved a high level of accuracy.

Various models, including DT, NB, kNN, and Extra Trees, have been applied with majority voting to defend against DDoS attacks [28]. Yoo et al. [29] proposed a hybrid model that combines a random forest and a DL model to classify files as either malware or benign. By employing hybrid majority voting rules, they achieved a significant improvement in DR. LSTM-based AE and one-class SVM methods have been used for encoding and classification in SDN networks, achieving high accuracy rates [30]. Ensemble approaches based on k-means++ and random forest have also

demonstrated excellent precision and recall in detecting attacks on SDN datasets [31]. Transforming SDN network traffic tabular data into image data and employing modified CNN models have also yielded high accuracy rates [32]. Another studies ([33], [34]) use the widely used and benchmark UNSW-NB15 dataset, achieving significant results with DBN and SVM, respectively.

While all the methods mentioned make valuable contributions to the existing literature, to the best of our knowledge, there is no study in the literature that systematically assesses and compares anomaly detection across datasets generated in diverse network environments, considering factors such as class imbalance, hyperparameter optimization, and feature selection. To address these challenges in NIDS, this study evaluates wired, wireless, and SDN networks in terms of feature selection, class imbalance problems, hyperparameter optimization, and binary classification tasks. By considering these factors, the goal is to efficiently detect network anomalies in different network environments. Table 2.1 shows that the proposed methodologies yield superior classification outcomes compared to the literature.

Table 2. 1 Comparison of existing works using the UNSW-NB15, AWID, and InSDN datasets.

Ref	Dataset	Method	Prec	Rec	F1	Acc	Year
[19]	UNSW-NB15	LDA and Random Tree	0.861	0.865	-	86.46%	2018
[33]	UNSW-NB15	Improved DBN	-	-	-	86.49%	2020
[34]	UNSW-NB15	SVM	-	-	-	85.99%	2019
[8]	UNSW-NB15	ABC-LR	-	-	-	85.99%	2019
This study	**UNSW-NB15**	**Random Forest (with Bayesian opt.)**	- **0.93**	- **0.94**	0.8826 **0.9356**	88.25% **92.89%**	2023 **-**

[11]	AWID	Random Forest	0.96	0.96	0.95	99.106%	2018
[7]	AWID	GP	0.79	0.78	0.78	-	2018
[7]	AWID	Karoo-GP	0.82	0.79	0.80	-	2017
[22]	AWID	DNN based on Ladder Network	-	-	-	99.28%	2019
[35]	AWID	Hunger Games Search and Remora Optimization	0.9976	0.994	0.9958	99.16%	2022
This study	**AWID**	**Random Forest (with Bayesian opt.)**	**0.99**	**0.99**	**0.997**	**99.95%**	**-**
[28]	InSDN	V-NKDE	0.998	0.998	0.998	99.84%	2021
[30]	InSDN	LSTM-AE-OC-SVM	0.93	0.93	0.93	90.50%	2020
This study	**InSDN**	**AE+ Random Forest (with Bayesian opt.)**	**0.99**	**0.99**	**0.9998**	**99.96%**	**-**

Acc= Accuracy, Ref= Reference, F1= F1-score, Prec=Precision, Rec=Recall.

2.3 Materials and methods

In this research, three publicly available datasets are used: AWID, UNSW-NB15, and InSDN. The AWID dataset focuses on wireless network attacks and contains 155 features. The training set has 1,795,575 instances, and the testing set has 575,643 instances. The UNSW-NB15 dataset includes wired network attacks, and it contains various attack categories along with normal traffic. The InSDN dataset consists of SDN traffic data and includes attacks targeting different layers and SDN-specific attacks. It contains 77 features and 343,939 instances for normal and attack traffic.

Before conducting classification experiments, preprocessing steps have been performed on the AWID dataset to transform it into an efficient format. The UNSW-NB15 dataset does not require preprocessing. In the InSDN dataset, non-contributing features have been removed. To address the class-imbalance problem, oversampling techniques such as SMOTE, SMOTETomek, and Adasyn have been applied. Feature selection has been carried out using XGBoost to identify the most relevant network traffic features. Bayesian optimization has been utilized to optimize the hyperparameters of existing ML algorithms as well as the proposed technique. Finally,

evaluation metrics have been computed to assess the performance of the classification methods employed in the study.

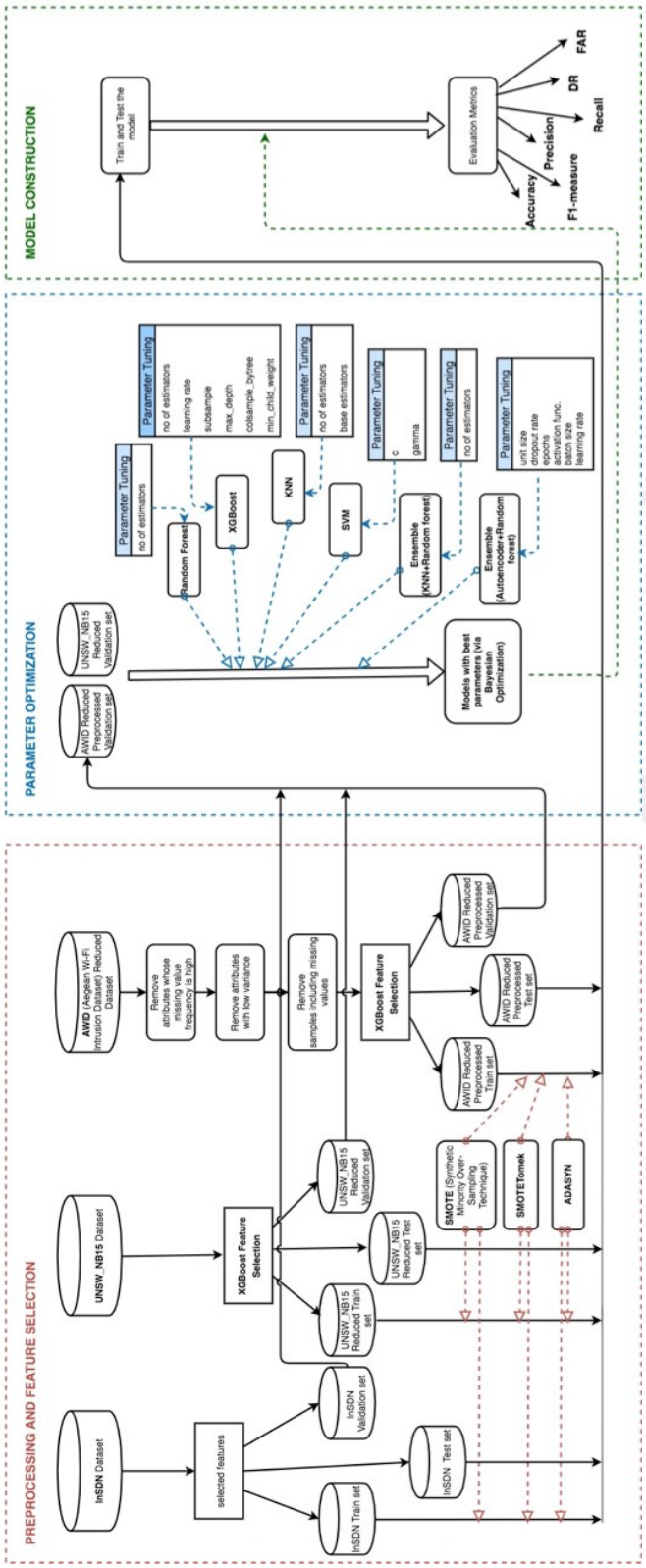

Figure 2. 1 A schematic representation of our methodology: preprocessing and feature selection, parameter optimization, and model construction.

2.3.1 Dataset and preprocessing

Data preprocessing constitutes a crucial and fundamental step in data mining, and it encompasses converting raw data into a format that is efficient and relevant for analysis. In the context of this study, data preprocessing techniques have been applied to the AWID, InSDN, and UNSW-NB15 datasets.

The AWID dataset contains 155 attributes, yet not all play a role in training the model. Within the AWID dataset, data varies in value and type—ranging from discrete and continuous to symbolic—creating a wide-ranging value spectrum. These diverse data characteristics pose a challenge for classifiers to accurately understand the underlying details. Hence, an essential step in classification is the pre-processing phase to navigate this complexity. In literature, some researchers ([7], [11], [22], [35]) mentioned in the comparison table (Table 2.1) that they have used a reduced version of the AWID dataset and have applied different preprocessing strategies. Firstly, specific features that do not affect variance and have a high number of missing values have been excluded from all studies. However, the exclusion of features varies from one study to another based on their specific criteria and the methods used for feature selection. Secondly, they have applied different strategies to samples containing missing values. Vaca et al. [11] have removed missing values with the most occurring values. Ran et al. [22] have replaced missing values with zeros. Kumar et al. [35] also used replacing missing values. In this study, the AWID dataset has been imported into SPSS[1], and specific features that have no discriminatory power and no effect on variance have been excluded based on their feature-frequency specifications. Features with a large number of missing values have been removed from the dataset. Replacing or deleting missing values has both advantages and drawbacks. Given that only two percent of the original data contains missing values. and to avoid errors or biases that may arise from "replacing" missing values, "deleting" was preferred in this study. Consequently, any remaining samples with missing values have been eliminated. As a result of these preprocessing methods, 20 out of the original 155 features from the AWID dataset have been retained.

Similarly, irrelevant features such as destination and source IP addresses, flow IDs, and timestamps were removed from the InSDN dataset during the preprocessing

[1] IBM Corp. IBM SPSS Statistics [online]. Website https://www.ibm.com/support/pages/downloading-ibm-spss-statistics-25 [accessed 01 Semptember 2022].

step. These attributes were eliminated as they were deemed irrelevant for the classification task. With the application of these preprocessing techniques, the datasets were cleansed and rendered suitable for subsequent classification and analysis.

The UNSW-NB15 dataset offers distinct testing and training datasets. The training set comprises 175,341 samples, with 119,341 labeled as "abnormal" and 56,000 labeled as "normal." Similarly, the testing set includes 82,332 samples, with 37,000 labeled as "normal" and the remaining 45,332 labeled as "abnormal" traffic samples. Hence, with the UNSW-NB15 dataset containing categorical features, the one-hot encoding technique is employed to convert these categorical features into numerical values. After encoding, the three categorical features, including 'service,' 'state,' and 'proto' in the UNSW-NB15 dataset, contribute to an increased total of 45 to 196 features.

To minimize the impact of different scales across features and reduce computational costs and training time, normalization techniques are applied to all datasets. Several normalization strategies exist in the literature, such as the Max-Abs scaler, the Standard scaler, and the Min-Max scaler. Considering the sparsity analysis of the datasets, the standard scaler was preferred.

2.3.2 Evaluation metrics

A critical stage in model development, the evaluation of ML algorithms involves assessing the model's performance using various evaluation metrics, which are obtained from the confusion matrix (Table 2.2). Although accuracy (Equation (2.1)) is commonly used, it is essential to consider additional metrics for a comprehensive evaluation of the model's performance. When dealing with imbalanced datasets, such as intrusion detection and others with a greater proportion of normal samples than abnormal samples, accuracy may fail to provide a comprehensive understanding. Metrics such as precision, F1-score (Equation (2.4)), recall, FAR (Equation (2.3)), and DR (Equation (2.2)) become critically important in such situations.

The metric of precision assesses the model's capability to accurately detect positive instances by dividing the number of predicted positive samples by the number of actual positive samples.

Recall, also known as true positive rate (TPR) or sensitivity, measures the percentage of real positive samples that are correctly identified.

The F1-score, a combination of recall and precision, provides an integrated performance evaluation that considers both aspects. The DR is the ratio of actual positive samples to the total number of positive samples. The FAR is calculated by dividing the number of false positive samples by the overall count of actual negative samples, indicating the model's tendency to incorrectly label negative instances as positive.

In addition to accuracy, in this study, other evaluation metrics such as precision, F1-score, recall, DR, and FAR have been used to assess the performance of an algorithm for ML in a more comprehensive manner.

Table 2. 2 Confusion Matrix

	Predicted Negative	Predicted Positive
Actual Negative	True Negatives (TN)	False Positives (FP)
Actual Positive	False Negatives (FN)	True Positives (TP)

$$\text{Accuracy} = \frac{\text{T P} + \text{T N}}{(\text{T N} + \text{FP} + \text{FN} + \text{T P})} \quad \textbf{(2.1)}$$

$$\text{Detection Rate (DR or T PR)} = \frac{\text{T P}}{(\text{FN} + \text{T P})} \quad \textbf{(2.2)}$$

$$\text{False Alarm Rate (FPR)} = \frac{\text{FP}}{(\text{T N} + \text{FP})} \quad \textbf{(2.3)}$$

$$\text{F1 score} = \frac{2 * \text{T P}}{(2 * \text{T P} + \text{FP} + \text{FN})} \quad \textbf{(2.4)}$$

2.3.3 Class imbalance problem

When the minority class in a dataset contains fewer samples than the majority class, traditional ML algorithms may struggle to perform well. This study employs oversampling rather than undersampling to resolve this issue and prevent the loss of informative minority class samples.

The SMOTE method has been applied as the initial oversampling technique on the preprocessed datasets. SMOTE is commonly used to fix class imbalance by creating synthetic data points for the minority class, using nearest neighbors. This balances class distribution and boosts classification accuracy and F1-measure.

In addition to SMOTE, the SMOTETomek algorithm has been applied to all datasets. SMOTETomek combines the capabilities of SMOTE for generating synthetic data with the Tomek Link undersampling method. Tomek Links, which are pairings of samples belonging to distinct classes that are in close range to one another, can be eliminated to enhance the differentiation between classes.

Furthermore, the Adasyn algorithm has been implemented as the final step of oversampling. Adasyn is an enhanced version of SMOTE that adds small random values to the synthetic samples generated by SMOTE. This improves the performance of the classification model by increasing the diversity and realism of the synthetic samples.

By using oversampling methods like SMOTE, SMOTETomek, and Adasyn, the problem of imbalanced class distribution can be reduced. These algorithms help the classification models learn better from the minority class samples, leading to improved accuracy and F1-measure for classification tasks.

2.3.4 Feature selection via extreme gradient boosting (XGBoost) algorithm

In this study, the XGBoost algorithm has been applied for feature selection in the network intrusion detection model construction. Feature selection plays a crucial role in reducing computation costs and improving the classification performance of the model.

XGBoost assigns relative relevance values to features based on their importance in making critical decisions on DT. The more a feature is used in decision making, the higher its importance score. By applying a threshold, which has been determined based on the point where feature importance scores decline rapidly, the top-ranked features can be selected.

Figure 2.2 and Figure 2.3 provide visual representations of the importance scores of each feature for the AWID and the UNSW-NB15 datasets. These figures present the relative importance scores of the selected features based on their respective weights, where the cumulative importance scores of all features amount to 1. For the AWID dataset, experiments encompassed 20 features ranging from "wlan.fc.subtype" to "radiotap.channel.type.cck," as well as 12 features covering "wlan.fc.subtype" to "wlan.fc.frag," as indicated in Table A.1. Similarly, experiments were conducted for the UNSW-NB15 dataset, encompassing 37 features spanning from "sttl" to "service=ftp," and 20 features from "sttl" to "dloss," as indicated in Table A.2. Feature selection has

not been applied to the InSDN dataset because it produces efficient results in its current form [36].

The feature selection step helps to identify the most relevant and informative features, which can lead to improved classification performance. By selecting critical attributes, the model can efficiently reduce computational costs while simultaneously improving its accuracy in identifying network intrusions.

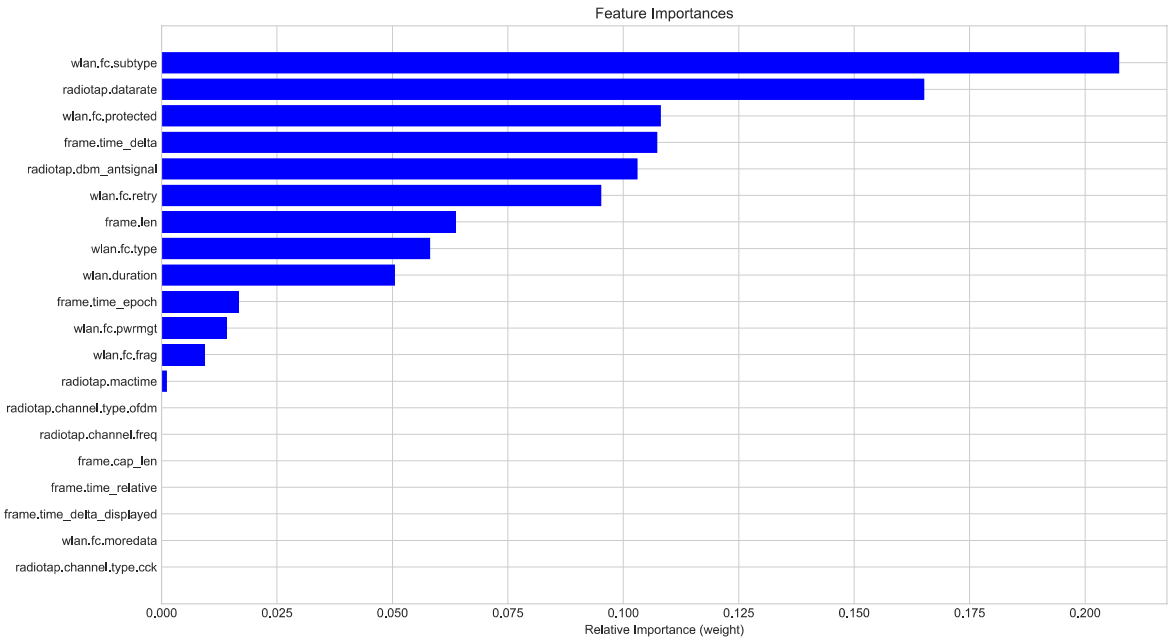

Figure 2. 2 The top 20 features with higher relative importance in the AWID dataset. The Y axis corresponds to the names of these features, and the X axis corresponds to the relative importance values of the corresponding features.

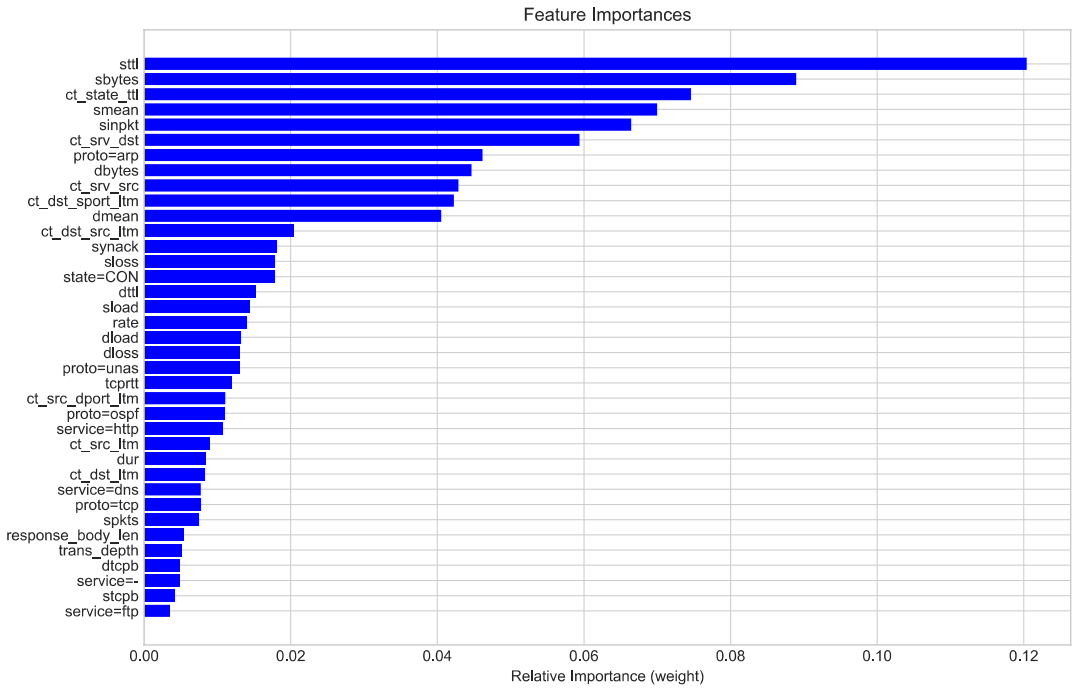

Figure 2. 3 The top 37 features with high relative importance in the UNSW-NB15 dataset. The Y axis corresponds to the names of these features, and the X axis corresponds to the relative importance values of the corresponding features.

2.3.5 Feature extraction via AE

Figure 2.4 depicts the AE, an unsupervised DL model designed to handle high-dimensional data. The AE compresses the input data into a bottleneck hidden layer, representing the encoded input data. The decoder component of the AE reconstructs the original input data by leveraging the encoded data and minimizing the reconstruction loss. During the training process, AEs aim to minimize the reconstruction error. In this particular context, the AE model has been trained using the training set and validated using the validation set. Following the Bayesian optimization process, a random forest classifier has been applied to the best encoded samples (compressed data shown in Figure 2.4) obtained from the AE model.

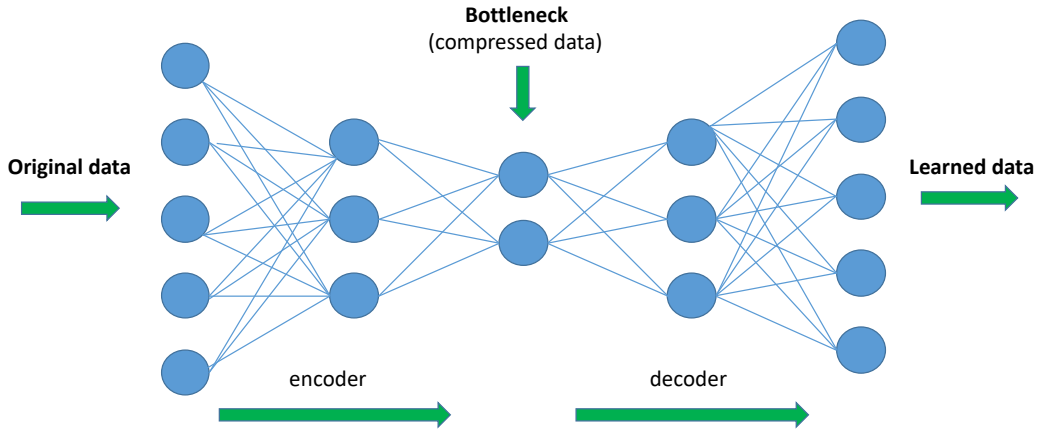

Figure 2. 4 Illustration of the autoencoder (AE) model.

2.3.6 Hyperparameter optimization using bayesian optimization

Optimizing parameters is a critical aspect of learning algorithms. While ML algorithms come with default parameters, tuning them is essential to achieve optimal performance. In this study, we explore various classification algorithms with different hyperparameters. To prevent overfitting, we employ validation sets instead of test sets. Stratified random sampling has been utilized to create balanced validation sets, which involve randomly sampling 30 percent of the imbalanced versions of the AWID and UNSW-NB15 training sets.

Table 2.3 presents the tuned hyperparameters for each classification algorithm along with the search approaches employed. For an SVM model, "c" and "gamma" are hyperparameters that require careful balancing between bias and variance. To find optimal values for these parameters, a randomized search strategy has been employed. The parameter named "n_estimators" refers to the total number of trees present in the forest when employing algorithms like Random Forest. In the XGBoost model, "learning_rate" is a hyperparameter that determines the weight adjustment of newly added trees.

Furthermore, the parameters "subsample", "max_depth", "colsample_bytree," and "min_child_weight" correspond to the following properties in the context of base

learners: the maximum depth allowed for each tree, the ratio of columns to consider when building each tree, the minimum required sum of instance weight in a child node, and the ratio of training instances to use during the construction of individual trees, respectively.

Table 2. 3 Ranges of classifier's hyperparameters used for Bayesian optimization.

Method	Parameter	Lowest	Highest
Random forest	n_estimators	10	500
Random forest	min_samples_leaf	1	5
Random forest	min_samples_split	2	10
Random forest	max_depth	1	150
SVM	c	0.001	1
SVM	gamma	0.01	1
XGBoost	n_estimators	10	500
XGBoost	subsample	0.3	1
XGBoost	max_depth	2	10
XGBoost	colsample_bytree	0.1	0.6
XGBoost	learning_rate	0.01	0.07
XGBoost	min_child_weight	1	5
AE	learning rate	10^{-8}	10^{-1}
AE	no. of hidden units	5	50
AE	dropout rate	0	0.5
AE	batch size	1	1024
AE	no. of epochs	1	50

2.4 Experiments and Results

The research methodology for this study comprises three primary phases. First, the contribution of feature selection using XGBoost has been evaluated in the classification problem. Second, the impact of different imbalance strategies on classification outcomes has been investigated. Third, the effectiveness of Bayesian hyperparameter optimization in classification has been assessed. Performance metrics have been compared across wired, wireless, and SDN networking flows.

For the classification task, several ML algorithms have been employed. The SVM algorithm has been used to evaluate IDS datasets with respect to the linearity aspect. Tree-based models, including random forest and XGBoost, have been utilized, which make use of if-then rules for classification. The k-nearest neighbor algorithm has

been applied both independently and in combination with random forests to create an ensemble classifier. These classification methods have been implemented using the scikit-learn library in Python.

When evaluating the performance of classification models using accuracy, it is important to address issues related to imbalanced datasets. The number of samples from different classes should not vary significantly. Additionally, in this study, other evaluation metrics such as F1 score, precision, recall, DR, and FAR have been utilized.

Table 2.4, Table 2.5, and Table 2.6 provide a summary of the performance results from the experiments with optimal hyperparameters obtained through Bayesian optimization. These experiments include ablation studies, that is, the effect of feature selection and different class imbalance strategies on classification results.

In summary, the experiments demonstrate the effectiveness of feature selection, imbalance strategies, and the Bayesian hyperparameter optimization in the classification of network intrusion detection datasets. It has been observed that the proposed intrusion detection systems for wired, wireless, and SDN networks perform well in terms of normal/anomaly detection when hyperparameters are optimized on the validation set and evaluated on the test set. Balanced versions of the datasets yield better performance compared to imbalanced versions. The best performance of NIDS has been reported in terms of accuracy, precision, recall, F1 measure, DR, and FAR. The results show that the optimum values for F1 measure, overall accuracy, DR, and FAR are high in all three datasets.

Table 2. 4 Binary classification with Bayesian optimization results on the AWID dataset. The bold ones present the best F1-measure and accuracy scores for the AWID dataset and its subsets.

Dataset		Model	Precision	Recall	F1-Score	Accuracy
	Imbalance	kNN	0.85	0.91	0.879	0.9798
		Random Forest	0.97	0.81	0.8828	0.9827
		SVM	0.28	0.99	0.4365	0.7943
		XGBoost	0.94	0.12	0.2128	0.9286
		kNN+RF	0.96	0.32	0.4799	0.9442
		AE+kNN	0.98	0.42	0.588	0.9526
	Balance	kNN	0.84	0.91	0.8736	0.9788
		Random	0.96	0.82	0.8845	0.9828

AWID (12 features)		Forest				
		SVM	0.43	0.43	0.4299	0.9083
		XGBoost	0.95	0.81	0.8744	0.9813
		kNN+RF	0.98	0.84	**0.9046**	**0.9857**
		AE+kNN	0.98	0.42	0.588	0.9526
AWID (20 features)	Imbalance	kNN	0.84	0.86	0.8499	0.9756
		Random Forest	0.97	0.66	0.7855	0.971
		SVM	0.63	0.63	0.63	0.9404
		XGBoost	0.97	0.49	0.6511	0.9577
		kNN+RF	0.97	0.65	0.7784	0.9702
		AE+kNN	0.98	0.46	0.6261	0.9558
	Balance	kNN	0.84	0.91	0.8736	0.9788
		Random Forest	0.99	0.99	**0.997**	**0.9995**
		SVM	0.43	0.43	0.4299	0.9083
		XGBoost	0.79	0.80	0.795	0.9668
		kNN+RF	0.98	0.82	0.8929	0.9842
		AE+RF	0.98	0.46	0.6261	0.9558

Table 2. 5 Binary classification with Bayesian optimization results on the UNSW-NB15 dataset. The bold ones present the best F1-measure and accuracy scores for the UNSW-NB15 dataset and its subsets.

Dataset		**Model**	**Precision**	**Recall**	**F1-Score**	**Accuracy**
UNSW-NB15 (37 features)	Imbalance	kNN	0.84	0.92	0.8782	0.8595
		Random Forest	0.82	0.99	0.8955	0.8732
		SVM	0.74	0.97	0.8469	0.803
		XGBoost	0.74	0.86	0.7955	0.7565
		kNN+RF	0.84	0.92	0.8781	0.8595
		AE+RF	0.81	0.99	0.891	0.8666
	Balance	kNN	0.86	0.91	0.8843	0.8689
		Random Forest	0.87	0.98	**0.9155**	**0.9017**
		SVM	0.84	0.91	0.8736	0.855
		XGBoost	0.99	0.52	0.6819	0.7328
		kNN+RF	0.94	0.68	0.7891	0.7999
		AE+RF	0.85	0.97	0.906	0.8892

UNSW-NB15 (20 features)	Imbalance	kNN	0.83	0.90	0.8636	0.8434
		Random Forest	0.82	0.99	0.8941	0.8711
		SVM	0.74	0.98	0.8469	0.803
		XGBoost	0.76	0.94	0.8405	0.8035
		kNN+RF	0.83	0.90	0.8636	0.8434
		AE+RF	0.81	0.99	0.891	0.8666
	Balance	kNN	0.83	0.91	0.8682	0.8478
		Random Forest	0.93	0.94	**0.9357**	**0.9286**
		SVM	0.89	0.85	0.8695	0.8596
		XGBoost	0.98	0.56	0.7127	0.7514
		kNN+RF	0.90	0.71	0.7938	0.7969
		AE+RF	0.86	0.97	0.9117	0.8965
UNSW-NB15 (all features)	Imbalance	kNN	0.84	0.92	0.8781	0.8595
		Random Forest	0.82	0.99	0.8923	0.8690
		SVM	0.75	0.99	0.8553	0.8146
		XGBoost	0.78	0.97	0.8647	0.8328
		kNN + RF	0.84	0.92	0.8782	0.8595
		AE + RF	0.81	0.99	0.891	0.8666
	Balance	kNN	0.86	0.91	0.8843	0.8689
		Random Forest	0.93	0.94	**0.9356**	**0.9289**
		SVM	0.96	0.84	0.896	0.893
		XGBoost	0.99	0.55	0.7071	0.7492
		kNN + RF	0.96	0.67	0.7892	0.8029
		AE + RF	0.83	0.98	0.8988	0.8785

Table 2. 6 Binary classification with Bayesian optimization results on the InSDN dataset. The bold ones present the best F1-measure and accuracy scores for the InSDN dataset and its subsets.

Dataset		Model	Precision	Recall	F1-Score	Accuracy
	Imbalance	kNN	0.99	0.99	**0.9998**	**0.9996**
		Random Forest	0.99	0.99	0.9971	0.9953
		SVM	0.99	0.99	0.9982	0.9970
		XGBoost	0.99	0.99	0.9966	0.9946

InSDN		kNN + RF	0.99	0.99	0.9998	0.9997
		AE + RF	0.99	0.99	**0.9998**	**0.9996**
	Balance	kNN	0.99	0.99	0.9967	0.9947
		Random Forest	0.99	0.99	0.9970	0.9941
		SVM	0.99	0.99	0.9987	0.9979
		XGBoost	0.99	0.86	0.9263	0.8901
		AE + RF	0.99	0.99	**0.9998**	**0.9996**
		kNN + RF	0.99	0.99	0.9967	0.9947

Table 2. 7 Best results obtained from the AWID, UNSW, and InSDN datasets based on different numbers of selected features.

Dataset	#SF	Method	Imb. strategy	Prec	Rec	FM	Acc	DR	FAR
AWID	12	voting ensemble (kNN+ RF)	SMOTE + Tomek link	0.98	0.84	0.9046	0.9857	0.9797	0.0151
AWID	20	RF	Adasyn	0.99	0.99	0.997	0.9995	0.9999	0.0171
UNSW-NB15	37	RF	Adasyn	0.87	0.98	0.9155	0.9017	0.8688	0.0464
UNSW-NB15	20	RF	Adasyn	0.93	0.94	0.9357	0.9286	0.9243	0.0678
UNSW-NB15	all	RF	Adasyn	0.93	0.94	0.9356	0.9289	0.9328	0.07597
InSDN	all	AE+ RF	SMOTE + Tomek link	0.99	0.99	0.9998	0.9996	0.9998	0.0012

Acc= Accuracy, #SF=no. of selected features, FM= F1-score, Prec=Precision, Rec=Recall.

Chapter 3

Network anomaly detection using deep autoencoder and parallel artificial bee colony algorithm-trained neural network

3.1 Motivation

Metaheuristics and ANNs are two powerful techniques in the field of artificial intelligence. Metaheuristics are optimization algorithms that can be used to solve complex problems where traditional methods may not work. They function by intelligently and efficiently investigating a vast search space, often drawing inspiration from natural phenomena such as evolutionary processes or swarm behavior. On the other hand, ANNs are a type of ML model that can learn complex patterns and relationships from data. They are inspired by the structure and function of the human brain and can be used for a wide range of applications, such as predictive analytics and classification problems. Combining these two techniques can lead to even more powerful AI solutions that can solve complex problems in a more efficient and effective way.

In the study, the ABC algorithm was used to optimize the weights of ANNs. The weights of a neural network determine how it processes input data and produces output data. Finding the optimal weights for a neural network is a challenging problem that can be solved using optimization algorithms like the ABC algorithm. This study was able to optimize the weights of the neural networks efficiently and effectively. The

coded model was tested on a network intrusion detection problem, encompassing both multi-class and binary-class classification scenarios.

3.2 Related Work

To enhance the clarity of the literature review in this study, we have organized it into three sub-sections.

3.2.1 ABC-ANN literature review

The ABC technique is utilized to estimate the bias and weight values of the neural network model by minimizing the mean square error between the target and the output. Numerous studies in the literature utilize ANN to address a wide range of problems, including the field of NIDS.

Ali et al. [37] employ ABC for both feature selection and ANN weight optimization in order to detect DDoS attacks. They use a back-propagation neural network architecture that feeds inputs and adjusts weights simultaneously. However, there are no implementation and performance evaluation results provided on any dataset.

Karaboga et al. [38] first suggest training an ANN using ABC and comparing its performance with other population-based algorithms. Ozturk et al. [39] propose a hybrid model that combines ABC and Levenberg-Marquardt (LM) algorithms for training an ANN model. Both studies evaluate their models using 3-Bit Parity, XOR, and 4-bit Encoder-Decoder problems. This research highlighted the potential of using the ABC algorithm as an optimization technique for training ANNs.

In training ANN, the results were consistent with earlier research, which suggested that the ABC approach outperformed the backpropagation algorithm ([40]). [41] combines ABC and Particle Swarm Optimization (PSO) algorithms to optimize the weights of a Multi Layer Perceptron (MLP) for predicting the heating and cooling loads of residential buildings using 768 samples. Anuar et al. [42] propose the ABC-ANN method for crime classification using a crime dataset with 128 attributes and 1994 instances. They evaluate the performance of ANN-ABC using only the accuracy metric.

Other studies include ANN trained by ABC for FRP-concrete bond strength evaluation [43] using 656 samples, forecasting blast-produced ground vibration [44] with 89 blasting events, determining the vibration period of reinforced concrete infilled

framed structures [45] with 4025 samples, and intrusion detection using a combination of fuzzy clustering, MLP, and ABC [46] on the NSL-KDD and CloudSim datasets.

[47] used the ABC-ANN model for intrusion detection and achieved 87% accuracy on the NSL-KDD dataset. However, their study did not focus on time and speed considerations or the use of a hybrid approach combining the DAE and the ABC-ANN model.

In summary, the ABC algorithm is used for training ANN models to avoid local minimum solutions. However, it suffers from long training times to find global solutions. Existing studies that use the ABC approach for ANN training are often trained on small datasets. To address these challenges, this study proposes a novel hybrid approach combining DAE and ANN models trained by an ABC algorithm in parallel with Bayesian hyperparameter optimization.

3.2.2 NIDS literature review

Anomaly detection, especially in NIDS, has remained a long-standing yet dynamically evolving research domain across various research communities for decades [48]. Most studies utilize ML and DL techniques and hybridize them with various techniques, such as fuzzy logic-based decision systems [49], to detect network anomalies.

Some studies employ different concepts to detect anomalies in network traffic data. One of these studies [50] uses concept drift to detect attacks in network flows by monitoring changes in the network traffic distribution or alterations in the characteristics of the network traffic. They use SVM classifiers for classification purpose and obtain significant results on the Testbed dataset, NSL-KDD, and CIDDS-2017.

Zhong et al. [51] introduce a novel anomaly detection framework that integrates multiple DL techniques, including the Damped Incremental Statistics algorithm for feature extraction from network traffic, the AE for assigning abnormal scores to network traffic, LSTM for classification, and a weighted method for obtaining the ultimate abnormal score. They use the Mirai dataset [52], and the HELAD algorithm demonstrates good adaptability and accuracy compared to other state-of-the-art algorithms. Another study [53] comprises two primary steps. Firstly, a DBN is employed for nonlinear feature extraction, automatically extracting features from the data while reducing its dimensionality. Subsequently, a lightweight LSTM network is utilized to classify the extracted features, thereby generating classification results. The

researchers tested their model on the KDD99 and CICIDS2017 benchmark datasets, obtaining satisfactory results.

Detecting abnormal patterns and attacks using graph-based anomaly detection is another area of focus. In a study conducted by Deng et al. [54], a Graph Deviation Network (GDN) approach based on graph neural networks (GNNs) has been proposed, yielding significant results in detecting anomalies and attacks in the sensor data of cyberphysical Systems.

In another study [55], addressing the issue of few-shot network anomaly detection involves proposing a novel type of GNN called GDN. These networks can utilize a limited number of labeled anomalies to enforce statistically significant deviations between normal and abnormal nodes in a network.

3.2.3 Metaheuristics on NIDS

In literature, metaheuristics are commonly used for different objectives, such as feature selection [56], [57], [58], and hyperparameter optimization. Only a few researchers have utilized metaheuristics with the aim of training neural networks and DL architectures [59]. [60] has constructed an NIDS model for training MLP using a hybrid metaheuristic that combines the ABC algorithm and the Dragonfly Algorithm (DA). This study has obtained significant results in terms of DR, FAR, and accuracy on different public network datasets. [61] has offered a method-based Chronological Salp Swarm Algorithm for the weight optimization of DBN for the detection of intrusions. It has performed results on the BoT-IoT dataset and the KDD cup dataset and obtained significant results.

[62] has developed a RaNN model whose hyperparameters are tuned by hybrid PSO with Sequential Quadratic Programming (SQP). In the study of Elmasry et al. [63], they have utilized PSO for both the purposes of feature selection and hyperparameter optimization. Subsequently, they have tested this pre-trained model using three DL algorithms: DNN, LSTM-RNN, and DBN. Kanna et al. [64], in the first stage, have applied feature selection by the ABC algorithm and hyperparameter optimization by Black Window Optimization (BWO) algorithms. Subsequently, it has applied Convolutional and LSTM neural networks for intrusion detection. Saif et al. [65] have tried to reduce the computation cost of intrusion detection systems for IoT-based healthcare systems via metaheuristic algorithms. It has employed algorithms such as Differential Evolution (DE), Genetic Algorithm (GA), and PSO, attaining substantial

outcomes on the NSL-KDD dataset. Another study [66] has employed a novel approach called the Horse Herd Optimization Algorithm (HOA) that mimics horse behaviors within a herd to select relevant features for detecting intrusions. It has obtained significant results on the CSE-CIC-IDS2018 and NSL-KDD datasets. On the other hand, the study by Malibari et al. [67] have employed a metaheuristic called quantum-behaved particle swarm optimization (QPSO) with the aim of optimizing the hyperparameters of the deep wavelet neural network (DWNN) model. This model is designed for constructing intrusion detection systems for secure, smart environments. Ponmalar et al. [68] has optimized CNN hyperparameters via a hybrid metaheuristic approach, which is a combination of both the whale optimization algorithm and the local search of the Tabu optimization algorithm.

In spite of the fact that all studies have different contributions to network intrusion detection research, they may have some limitations, such as higher computational complexities, longer training times, and lower DR. This study suggests overcoming the limitations mentioned above.

Table 3. 1 Related works that apply ABC-ANN and other metaheuristics.

Reference	Problem	Method	Optimized Components	Data size	ttime
[37]	DDos attack detection	ABC-ANN	weights	X	X
[38]	XOR, 3-Bit Parity and 4-Bit Encoder-Decoder	ABC-ANN	weights	X	X
[39]	XOR, 3-Bit Parity and 4-Bit Encoder-Decoder	ABC-LM-ANN	weights	X	X
[41]	Prediction of the heating and cooling loads of residential buildings	ABC-MLP PSO-MLP	weights	8 features 768 samples	X
[43]	FRP-concrete bond strength evaluation	ABC-ANN	weights	656 samples	X
[44]	Forecasting the blast-produce ground vibration	ABC-ANN	weights	89 samples	X
[46]	Intrusion detection for cloud computing	ABC-ANN	weights	41 features 7 million samples	
[60]	Network intrusion detection	ABC-DA-ANN	weights	(big data) UNSW-NB15, ISCX2012,KDD	X

[61]	Network intrusion detection	SSA-DBN	weights	Cup 99 NSL-KDD (big data) KDD cup, BoT-IoT	✓
[62]	IIoT network intrusion detection	PSO-SQP RaNN	hyper parameters	(big data) DS2OS, UNSW-NB15, ToN_IoT	X
[63]	Network intrusion detection	PSO,DNN, LSTM-RNN, and DBN	no. of features and hyper parameters	(big data) CICIDS201 and NSL-KDD	✓
[64]	Network intrusion detection	ABC, BWO, CNN, LSTM	no. of features and hyper parameters	(big data) NSL-KDD, ISCX-IDS UNSW-NB15, CSE-CIC-IDS2018	✓
[65]	Network intrusion detection	ABC-DA-ANN	no. of features	(big data) NSL-KDD	✓
[66]	Network intrusion detection	HOA	no. of features	(big data) NSL-KDD and CSE-CIC-IDS2018	X
[67]	Network intrusion detection	QPSO DWNN	hyper parameters	(big data) CICIDS2017	X
[68]	Network intrusion detection	WOA-Tabu CNN	hyper parameters	(big data) NSL-KDD, KDD-Cup99, UNSW-NB15	X
proposed method	**Network intrusion detection**	**DAE-ABC-ANN**	**weights**	**UNSW-NB15 and NF-UNSW-NB15-v2**	✓

ttime = training time.

3.3 Proposed Methods

This section comprehensively explores various aspects crucial to the development and implementation of an effective network intrusion detection system (NIDS) (illustrated in Figure 3.1). Beginning with feature extraction via DAE and feature selection via the XGBoost algorithm, it explains the critical processes of data preprocessing and feature engineering. Subsequently, the ABC algorithm and its adaptation to the ANN framework are detailed, highlighting the innovative approach taken to optimize model training. Moreover, the section discusses the significance of data vectorization and parallel computation on GPUs, shedding light on the computational strategies employed to enhance efficiency and scalability. Lastly, it addresses the utilization of Bayesian optimization, offering insights into the techniques employed for fine-tuning model parameters and maximizing classification performance. Through an in-depth examination of these key components, this section proposes a

DAE-based parallel ABC-ANN method, contributing to advancements in network intrusion detection methodologies.

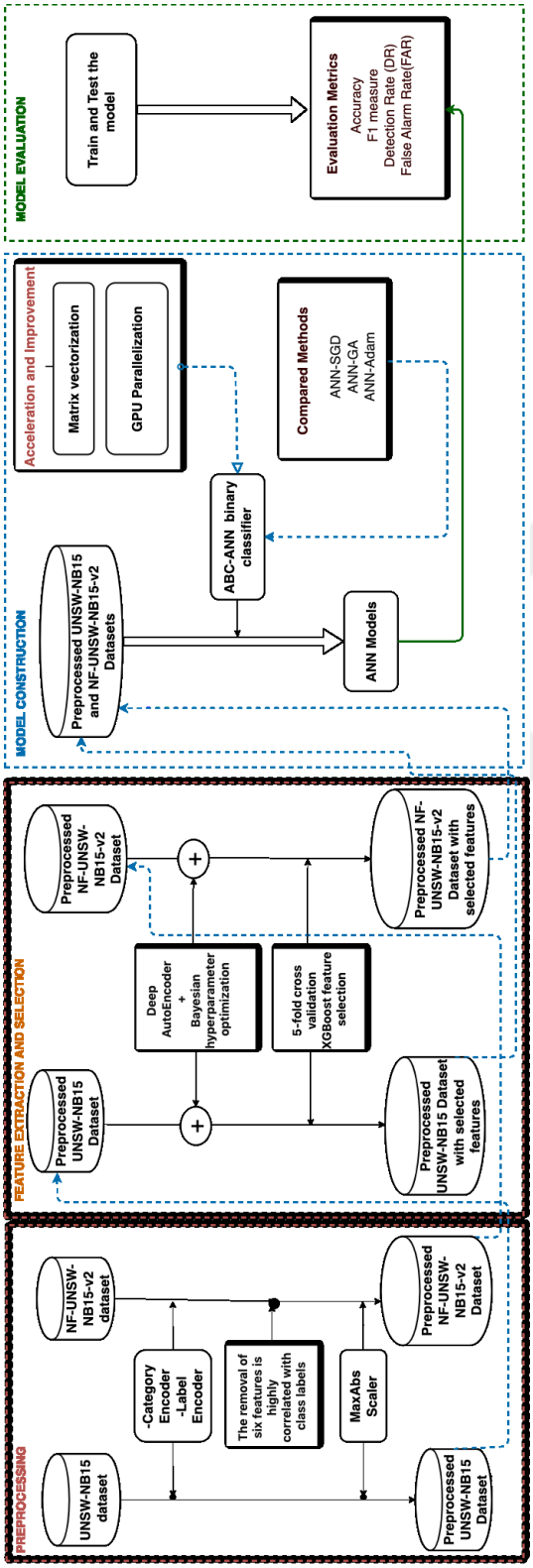

Figure 3. 1 Illustration of the proposed DAE-based ABC-ANN network intrusion detection methodology for binary classification, with the preprocessing, feature extraction and selection, model construction, and model evaluation processes highlighted in red, orange, blue, and green, respectively.

3.3.1 Feature extraction via deep autoencoder (DAE)

DAE is a form of DNN used to reduce dimension and extract attributes. The main purpose of a DAE is to discover a compressed representation of input data while minimizing information loss. This is done by training the network to reproduce the input in the output layer. Figure 2.4 shows a typical architecture of a DAE consisting of an encoder that converts input data to a compressed version and a decoder that recreates the original input from the encoded data. Encoders compress the data into a lower-dimensional space and effectively capture the most important features of the input in a non-linear way. In this study, a new encoded representation of the input data is extracted using DAE.

This investigation was utilized via Bayesian optimization. Using the domain ranges specified in Table 3.4 for the DAE section, hyperparameters were optimized for both datasets, and the optimal parameters in Table 3.5 were obtained. Encoded data capturing the most important features of original data has been combined with original data for further analysis. This consolidated data set was then used as input for the feature selection step (Figure 3.3, Figure 3.4, Figure 3.5, and Figure 3.6).

By using a DAE for feature extraction, this study aims to obtain a more compact and informative representation of the data, which can potentially improve the performance of following analysis tasks such as classification or anomaly detection. The encoded features can help reduce the dimensionality of the data while retaining important information, thus aiding in more efficient and effective feature selection processes.

3.3.2 Feature selection via xgboost algorithm

Ensembles of DT approaches, such as XGBoost, have the advantage of being able to automatically generate feature importance scores from a trained model. These scores indicate the relative importance or usefulness of each feature in the model's decision-making process. Features that are consistently used to make crucial decisions in the ensemble of DT will have higher importance scores.

In this study, it is aimed to find the optimal set of features from the NF-UNSW-NB15-v2 and the UNSW-NB15 datasets by considering both the original features and the encoded features obtained from the DAE. To achieve this, it utilizes the feature importance scores provided by XGBoost.

To select the most informative features, it employs 5-fold cross-validation and calculates the feature importance scores using XGBoost. Then, it examined the F1 scores and accuracy scores for different combinations of selected features, including both the original and encoded features (as shown in Figure 3.3, Figure 3.4, Figure 3.5 and Figure 3.6).

From these analyses, it is evident that the best results in terms of accuracy and F1 score are obtained when the encoded features and original features are concatenated. In the UNSW-NB15 dataset, only the top 30 features are selected based on the XGBoost feature importance scores, while in the NF-UNSW-NB15-v2 dataset, only 40 features are selected. Therefore, the experiments are carried out using subsets of 30 and 40 selected features, respectively. This approach allows us to focus on the most relevant features and potentially improve the accuracy of the findings.

Table 3. 2 Selected 30 features using the 5-fold cross-validation XGBoost method obtained from a combination of the UNSW-NB15 original features and encoded features. Sum of all importance scores equal 1.

Feature name	Format	Feature importance scores
sttl	integer	0.37267
ct_srv_dst	integer	0.06874
ct_dst_src_ltm	integer	0.04268
encoded f42	float	0.03717
synack	float	0.02834
sbytes	integer	0.02769
ct_state_ttl	integer	0.02617
encoded f16	float	0.02571
service=-	categorical	0.02395
ct_srv_src	integer	0.02207
ct_dst_sport_ltm	integer	0.019221
encoded f43	float	0.01834
encoded f15	float	0.01737
smean	integer	0.01649
encoded f48	float	0.01343
proto=tcp	categorical	0.01192
encoded f24	float	0.01173
encoded f52	float	0.01146
encoded f49	float	0.01140

dbytes	integer	0.01131
dmean	integer	0.00886
encoded f18	float	0.00838
encoded f27	float	0.00788
encoded f26	float	0.00782
service=http	categorical	0.00700
encoded f21	float	0.00691
service=dns	categorical	0.00642
service=ftp	categorical	0.00619
encoded f32	float	0.00482
encoded f40	float	0.00482

Table 3. 3 Selected 40 features using the 5-fold cross-validation XGBoost method obtained from a combination of the NF-UNSW-NB15-v2 original features and encoded features. Sum of all importance scores equal 1.

Feature name	Format	Feature importance scores
MIN_IP_PKT_LEN	integer	0.67513
TCP_WIN_MAX_IN	integer	0.18188
SHORTEST_FLOW_PKT	integer	0.02189
DNS_QUERY_TYPE	integer	0.01588
LONGEST_FLOW_PKT	integer	0.01269
encoded f29	float	0.00533
RETRANSMITTED_OUT_BYTES	integer	0.00418
SERVER_TCP_FLAGS	integer	0.00395
L7_PROTO	float	0.00391
PROTOCOL	integer	0.00390
encoded f21	float	0.00380
encoded f22	float	0.00363
TCP_FLAGS	integer	0.00359
OUT_BYTES	integer	0.00296
encoded f17	float	0.00289
NUM_PKTS_UP_TO_128_BYTES	integer	0.00270
RETRANSMITTED_OUT_PKTS	integer	0.00249
encoded f13	float	0.00223
OUT_PKTS	integer	0.00220
encoded f5	float	0.00219
FTP_COMMAND_RET_CODE	integer	0.00205

CLIENT_TCP_FLAGS	integer	0.00192
DST_TO_SRC_SECOND_BYTES	float	0.00190
SRC_TO_DST_AVG_THROUGHPUT	integer	0.00182
encoded f12	float	0.00172
IN_PKTS	integer	0.00170
TCP_WIN_MAX_OUT	integer	0.00164
encoded f10	float	0.00139
encoded f4	float	0.00137
DST_TO_SRC_AVG_THROUGHPUT	integer	0.00133
encoded f14	float	0.00132
NUM_PKTS_128_TO_256_BYTES	integer	0.00127
SRC_TO_DST_SECOND_BYTES	integer	0.00126
RETRANSMITTED_IN_BYTES	integer	0.00124
encoded f9	float	0.00121
encoded f3	float	0.00121
IN_BYTES	integer	0.00116
encoded f8	float	0.00110
encoded f18	float	0.00107
encoded f11	float	0.00107

3.3.3 Artificial bee colony (ABC) algorithm

The ABC algorithm operates by simulating the behavior of honey bees. In the ABC algorithm, each food source corresponds to a potential solution for the optimization problem, and the quantity of food source indicates the fitness or quality of the solution.

The ABC algorithm consists of three main phases: the employed bee phase, the scout bee phase, and the onlooker bee phase. These phases collectively form an iterative process to search for optimal solutions.

In the employed bee phase, there are the same number of employed bees as there are food sources. Each employed bee examines a new food source in the neighborhood of its current food source. If the quantity (fitness) of the new source is higher than that of the previous source, the employed bee updates its memory by recording the new food source and forgets the previous one. The employed bees then

perform a dance within the colony to communicate the quantity and quality of their food sources.

In the onlooker bee phase, the onlooker bees observe the dance of the employed bees and choose their food sources based on the quality of the food source. The higher the quantity and quality of a food source, the more likely it is to be chosen by the onlooker bees. Each onlooker bee assesses a new food source in the neighborhood of the chosen food source, similar to what the employed bees do.

In the scout bee phase, abandoned food sources that have not been improved for a certain number of iterations are identified, indicating that they are not promising solutions. These abandoned food sources are replaced with new and unexplored food sources found by scout bees, which explore new areas of the search space.

These three stages are repeated iteratively until requirements of the optimization problem and the termination criteria are met. The ABC algorithm aims to find the optimal solution by continuously exploring the search space based on the information shared among employed bees, scout bees, and onlooker bees. Through this iterative process, the algorithm can efficiently search for high-quality solutions in the optimization problem domain.

3.3.4 Artificial neural networks (ANN)

ANN models are computational systems inspired by the neural structure of the human brain, aiming to replicate the information processing mechanisms of biological systems [69]. These models consist of interconnected nodes, called neurons, organized into layers, enabling them to learn and adapt from data. Data is received by the input layer and propagated through weighted connections to hidden layers, ultimately generating an output. Throughout the training process, the network adjusts these weights based on the provided dataset, enhancing its predictive or classification capabilities. The proposed neural network structure, as depicted in Figure 3. 3, consists of an input layer where each neuron represents a feature from the intrusion dataset along with a bias value. The activation function is applied for activating all weights. The hidden layer neurons and connections between the input and output layers imitate and simulate the structure of the human brain.

The effectiveness of ANNs lies in their ability to extract hierarchical features and perform nonlinear mapping, enabling them to capture intricate relationships within data. In this study, to address this capability, the ABC algorithm is employed during the

training phase of ANNs. This integration aims to prevent the occurrence of local minima and explore high-performance solutions within the solution space, thereby enhancing the robustness and effectiveness of the ANN model for various optimization tasks.

3.3.5 Adaptation of ABC to ANN for binary classification task

In this study, the ABC algorithm, which is a population-based optimization technique, is customized for use with Artificial Neural Networks (ANNs) to optimize the biases and weights. The original ABC algorithm's primary drawback is its long training times, especially when attempting to locate global solutions. To tackle this issue, this study suggests utilizing a vectorized and parallelized ABC-ANN algorithm. This proposed approach combines the advantages of the ABC algorithm with parallel computing techniques, effectively expediting the training process (detailed mention in Section 3.3.7).

The proposed neural network structure, as depicted in Figure 3.2, consists of an input layer where each neuron represents a feature from the intrusion dataset along with a bias value. The sigmoid function (Equation 3.1) is used to activate all weights. Connections between the hidden layer neurons, the input layers, and the output layer imitate and simulate the human brain structure. Equation 3.2 calculates the values of the hidden layers to produce the probability value in the subsequent step. The output layer produces binary outcomes according to Equation 3.3, where 0 is used for normal and 1 is used for attack, using the probability function shown in Equation 3.4.

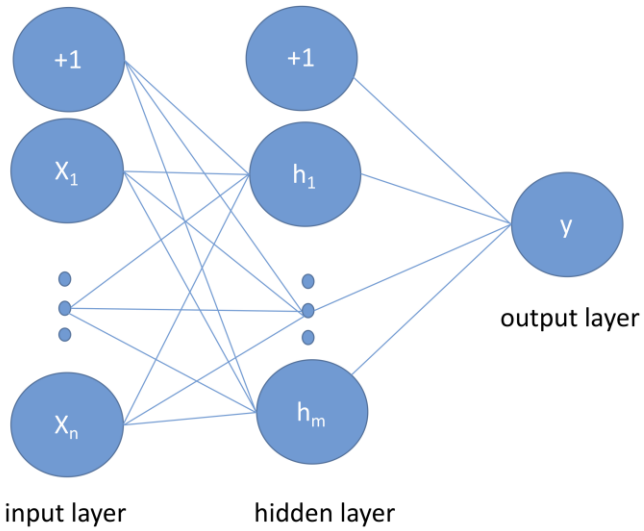

Figure 3. 2 Illustration of typical ANN for binary classification with a single hidden layer architecture.

$$\sigma\,(\,x) = \frac{1}{e^{(-x)}} \tag{3.1}$$

$$h_i = \sigma\,(\,x_i w'_{1i} + x_2 w'_{2i} + \cdots + x_n w'_{ni} + w'_{bi}) \tag{3.2}$$

$$y = \sigma\,(\,h_1 w''_{11} + h_2 w''_{21} + \cdots + h_m w''_{m1} + w''_{b1}) \tag{3.3}$$

$$p = \begin{cases} 1, & if\ y \geq 0.5 \\ 0, & otherwise \end{cases} \tag{3.4}$$

The proposed parallel ABC-ANN algorithm (Algorithm 1) combines the ABC optimization technique with ANN to create an effective classification method, termed the ABC-ANN classification method. It combines the collective and global search intelligence of bees to optimize the weight parameters of an ANN for classification tasks. Initially, the algorithm calculates the total number of weights and biases using the formula (presented in line 1 of Algorithm 1): D = (N + 1) × HLS + (HLS + 1) in the ANN model based on the number of neurons in the input and hidden layers, as well as the output layer. In the formula, N represents the number of neurons in the input layer and includes a bias term, while HLS denotes the number of neurons in the hidden layer. This sets the dimensionality of the solution space for the ABC algorithm.

Following the generation of the weight matrix, a matrix of food sources, defined by dimensions PxD, is generated for the solution space (Algorithm 2). Here, P represents the number of food sources (solutions), and D refers to the dimensionality of each solution. The fitness values of these solutions are computed based on their performance in the classification task (line 4).

In this study, two versions of the fitness function were implemented, taking into account the necessary adjustments for optimization across two different network datasets. The UNSW-NB15 dataset has a low class imbalance ratio, thus the Mean Absolute Error (MAE) (Algorithm 3) was optimized. On the other hand, the NF-UNSW-NB15-v2 dataset exhibits a significantly high class imbalance ratio; therefore, the F1score (Algorithm 4) was prioritized for optimization.

Taking into account the classification results from the output layer (Algorithm 5), F1 or accuracy scores are calculated for each food source, and then the fitness values are computed by averaging them.

Employed bees perform local searches around food sources (Algorithm 7), and if a new solution offers improvement, it replaces the old one. Onlooker bees select food sources based on their fitness values (Algorithm 8), conducting searches preferentially around better-performing solutions. If a solution's limit counter exceeds a predefined threshold, indicating no improvement, a scout bee generates a new solution (Algorithm 9). The best solution found represents the optimal or near-optimal set of weights for the ANN (Algorithm 10). The algorithm iterates over the search process until the maximum number of evaluations (MEN) is reached (line 7-31 in Algorithm 1).

By combining the exploration capabilities of the ABC algorithm with the learning and optimization capabilities of ANN, the proposed parallel ABC-ANN algorithm aims to find optimal weight values for the neural network to accurately detect network anomalies.

Algorithm 1: The Proposed Parallel ABC-ANN binary classification method
Input: Input matrix X_{MxN} , target $\vec{y}_M$, number of food sources *P*, maximum evaluation number *MEN,* position of the food sources W_{PxD} , upper bound *ub*, lower bound *lb,* limit, modification rate *MR*, hidden layer size *HLS* **Output:** 1: $D \leftarrow (N + 1) \times HLS + (HLS + 1)$ 2: *GENERATE_FOOD_SOURCES()*

```
3: W' ← W
4: f⃗it ← CalcFit(W)
5: τ⃗ ← zeros(P)                          ▷ P-dimensional zero vector
6: evaNumber ← 0
7: while evaNumber < MEN do
8:     SendEmployedBees()                                  ▷ Algorithm 6
9:     sf⃗it ← CalcFit(W')
10:    ind⃗ ← sf⃗it > f⃗it
11:    rind⃗ ← sf⃗it ≤ f⃗it
12:    τ⃗ [ind⃗] ← 0
13:    W[ind⃗] ← W'[ ind⃗]
14:    f⃗it [ind⃗] ← sf⃗it [ind⃗]
15:    τ⃗ [rind⃗] ← τ⃗ [rind⃗] + 1
16:    CalcProbabilities()                                 ▷ Algorithm 7
17:    SendOnlookerBees()                                  ▷ Algorithm 8
18:    sf⃗it ← CalcFit(W')                                 ▷ Algorithm 4
19:    for i ← 0 : P do
20:        temp ← tmpID⃗[i]
21:        if sf⃗it [i] > f⃗it [temp] then
22:            τ⃗ [temp] ← 0
23:            W[temp, :] ← W'[i, :]
24:            f⃗it [temp] ← sf⃗it [i]
25:        else
26:            τ⃗ [temp] ← τ⃗ [temp] + 1
27:        end if
28:    end for
29:    SendScoutBee()                                      ▷ Algorithm 9
30:    MemorizeBestSource()                                ▷ Algorithm 10
31: end while
32: return gpar⃗                                            ▷ return global
parameters
```

Algorithm 2: Create Food Source Positions

```
1: procedure GENERATE_FOOD_SOURCES
2: for x ← 0 : P do
3:     for y ← 0 : D do
4:         W[x,y] ← rand(0, 1) × (ub – lb) + lb
5:     end for
6: end for
7: end procedure
```

Algorithm 3: Calculate Mean Absolute Error based Fitness Function

```
1: procedure CALC FIT (φ )
2:     p_s ← CALCOut putLayer(φ )
```

```
3:    ε← absolute(p_s - y_M)
4:    f ← mean(ε, axis = 0)
5:    evaNumber ← evaNumber + len(f)
6:    return f/ (1+f)
7: end procedure
```

Algorithm 4: Calculate F1-score based Fitness Function (binary classification)

```
1: procedure CALC FIT (φ )
2:    p_s ← CALCOut putLayer(φ )
3:    p_s ← round (p_s)
4:    f ← F1 score(y_M, p_s)
6:    evaNumber ← evaNumber + len(f)
7:    return f
8: end procedure
```

Algorithm 5: Calculate Output Layer (binary classification)

```
1: procedure CalcOutputLayer(φ)
2:    M, N ← X.shape
3:    P ← φ.shape[0]
4:    p_s ← zeros(M, P)                          ▷ output neurons
5:    p_s ← p_s + φ[:,-1]                        ▷ bias addition
6:    for i ← 0 : HLS do
7:       W← φ[:, i × N : (i + 1) × N]^T
8:       b← φ[:, N × HLS+ HLS+ i ]^T
9:       z_i← σ(X.dot(W) +b)                     ▷ σ is sigmoid func
10:      p_s ← p_s + z_i*φ[:, FVS × HLS + i]
11:      end for
12:   end for
13:   p_s = σ (p_s)
14:   return p_s
15: end procedure
```

Algorithm 6: Calculate Probabilities

```
1: procedure CalcProbabilities(fit)
2:    maxfit ← max(fit)
3:    prob ← (0.9 × (fit / maxfit)) + 0.1
4:    return prob
5: end procedure
```

Algorithm 7: Employed Bee Phase

```
1: procedure SendEmployedBees
2:    for i ← 1 : P do
```

```
3:        arg⃗ ← rand(low = 0, high = 1, size = (D))
4:        ρ⃗ ← arg⃗ < MR                                  ▷ param to change
5:        η ← randint(1, P),     η ≠ i                   ▷ choose neighbour
6:        W'[i, :] ← W[i, :]
7:        vec ← W'[i, ρ⃗ ]
8:        vec ← vec + rand(–1, 1) × (vec – W[η, ρ⃗ ])
9:        vec[vec < lb] ← lb
10:       vec[vec > ub] ← ub
11:       W'[i, ρ⃗ ] ← vec
12:    end for
13: end procedure
```

Algorithm 8: Onlooker Bee Phase

```
1: procedure SendOnlookerBees
2:     i ← 0
3:     t ← 0
4:     while t < P do
5:         if rand(0, 1) < prob[i] then
6:             a⃗ ← rand(low = 0, high = 1, size = (D))
7:             ρ⃗ ← a⃗ < MR                              ▷ param to change
8:             η ← randint(1, P),    η ≠ i              ▷ neighbour
9:             W'[t, :] ← W[i, :]
10:            vector ← W'[t, ρ]
11:            vector ← vector + rand(–1, 1) × (vec – W[η, ρ⃗])
12:            tmpID⃗[t] ← i
13:            vector[vector < lb] ← lb
14:            vector[vector > ub] ← ub
15:            W'[t, ρ⃗] ← vector
16:            t ← t + 1
17:        end if
18:        i ← i + 1
19:        if i ≥ P then
20:            i ← 0
21:        end if
22:    end while
23: end procedure
```

Algorithm 9: Scout Bee function

```
1: procedure SendScoutBee
2:     index ← argmax(τ⃗)
3:     if τ⃗ [index] ≥ limit then
4:         for k ← 1 : D do
5:             W[index, k] ← lb + rand(0, 1) × (ub – lb)
6:             W'[index, k] ← W[index, k]
7:         end for
```

```
8:        fit[index] ← CalcFit(W[index, :])
9:        τ⃗ [index] ← 0
10:   end if
11: end procedure
```

Algorithm 10: Memorize Best Source

```
1: procedure MemorizeBestSource
2:     index ← argmax( fit⃗)
3:     if fit⃗ [index] > gmax then
4:         gmax ← fit⃗ [index]                    ▷ global maximum
5:         gpar⃗ ← W[index, :]                    ▷ global params
6:     end if
7: end procedure
```

3.3.6 Adaptation of ABC to ANN for multi-class classification task

The problem description and parameters of the proposed multi-class ABC-ANN algorithm are described below:

Problem: (Multi-class) Classification of network attacks and normal flows.

Cost function: Cost values for the confusion matrix.

Fitness Function: F1 macro or Accuracy.

Population size: The sum of the numbers of employed and onlooker bees in the population. (employed bees= onlooker bees).

Population range: Maximum and minimum values that weight corresponding to parameters in the character set can take.

Iteration: Maximum number of evaluations.

No. of food sources: The total number of solution sets stored in memory. Half the population size.

Limit: When the number of improvements to a solution exceeds the specified limit value, the solution is destroyed, and the algorithm again creates a random solution.

The proposed parallel ABC-ANN algorithm (Algorithm 11) integrates the ABC optimization technique with ANN to form a robust classification method, referred to here as the ABC-ANN classification method. This method leverages the collective search intelligence of bees to optimize the weight parameters of an ANN for classification tasks. In the initial step of the algorithm (Line 1), the total number of weights (D) in the ANN model is calculated using the formula (D = (N+1) * HLS + (HLS+1) * k), where (N) represents the number of neurons in the input layer plus a bias term, (HLS) denotes the number of neurons in the hidden layer, and (k) signifies the number of output classes. This calculation sets the dimensionality of the solution space that the ABC algorithm will explore. Line 2 performs one-hot encoding on the target vector $\vec{y}_M$, resulting in a matrix with dimensions M * k.

Following this, a P * D dimensional matrix of food sources is generated (Line 3) by invoking the "CreateFoodSources" function (Algorithm 2), where P corresponds to the number of food sources (solutions) and D refers to the dimensionality of each solution. Each row in this matrix represents a potential solution to the optimization problem, i.e., a specific set of weights for the ANN. The CalcFit function (Algorithm 13 and 14) is called on Line 5 to compute the fitness values of all P solutions based on their performance in the classification task, effectively evaluating the accuracy or efficacy of the ANN when using each set of weights. A P-dimensional zero vector $\vec{\tau}$ is initialized on Line 6 to track the limit values for each solution, representing the number of trials without improvement before a food source is abandoned.

The core of the algorithm (Lines 8-32) iterates over the search process until the maximum number of evaluations (MEN) is reached. Within this loop, employed bees perform local searches around each food source (Line 9), and the fitness of the newly found solutions is calculated (Line 10). If a new solution offers an improvement (Lines 11-16), it replaces the old one, and the corresponding limit counter is reset; otherwise, the counter is incremented. The probability of a food source being selected by an onlooker bee is calculated based on the fitness values of all solutions (Line 17), guiding the onlooker bees (Algorithm 8) to preferentially search around the better-performing solutions (Line 18). After onlooker bees have conducted their searches and new solutions have been evaluated (Line 19), a selection phase occurs (Lines 20-29), where improved solutions replace their predecessors, and limit counters are updated

accordingly. If a solution's limit counter exceeds a predefined threshold (limit), indicating it has not improved over several iterations, it is deemed exhausted, and a scout bee generates a new solution in its place (Line 30), facilitating global search and exploration.

Finally, the algorithm identifies the best solution found throughout the search process (Line 31), representing the optimal or near-optimal set of weights for the ANN to perform the given classification task. This methodology effectively combines the exploration and exploitation capabilities of the ABC algorithm with the predictive power of ANNs, aiming to find an optimal set of weight parameters that enhance classification performance.

Algorithm 2 generates and returns a weight matrix consisting of P solutions, each of which includes uniformly distributed random numbers between the lower (lb) and upper (ub) bound values. This matrix has dimensions of P * D, in accordance with the integers P and D taken as parameters. Algorithm 12 outlines the procedure for calculating the output layer values in a neural network with an input layer, a hidden layer, and an output layer. It is designed to handle a dataset with a specified number of samples (M) and features (N), across (P) different solutions. The process begins by determining the dimensions of the input dataset (X), extracting the number of samples (M) and features (N). It then identifies (P), the number of different solutions, from the first dimension of the matrix φ. A three-dimensional array (p_s) is initialized with dimensions (k * M * P), where (k) represents the number of output neurons (line 4). This array is to store the computed values for the output neurons across all samples and solutions. The algorithm adjusts the initial values of (p_s) (line 5-7) by adding a bias term from the φ matrix for each output neuron, incorporating the bias associated with each neuron.

For the hidden layer, the algorithm assigns weights w_{li} and biases b_{li} to each neuron. These weights and biases are extracted and transposed from φ, ensuring they match the required dimensions. The sigmoid activation function σ is applied to the sum of the dot product of the input data (X) with the neuron's weights and the neuron's bias, resulting in z_i (line 11). The output neurons' values in (p_s) are updated by adding the product of z_i and specific elements from φ, selected based on the current output neuron index and the hidden layer size. An exponential function is applied to (p_s), preparing it

for normalization (line 16). Normalization is achieved by computing the sum of (p_s) along its columns to obtain a vector $\vec{t}$ (line 17), which is used to normalize (p_s) by dividing each element by the corresponding element in $\vec{t}$ (line 18). This ensures that the output values for each sample, across all solutions, sum to 1, making them suitable for probability interpretation in classification tasks. The algorithm concludes by returning the normalized output neuron values p_s, representing the calculated values for the output layer of the neural network across all samples and solutions. This process accounts for the biases and the weighted contributions from the hidden layer, resulting in values appropriate for classification tasks.

In this study, two distinct versions (Algorithm 13 and Algorithm 14) of the fitness function are implemented. These two versions of the fitness function, which represents the f1 score-based and accuracy based fitness functions, (Algorithm 13 and Algorithm 14) commences by employing the "CalcOutputLayer" function (Algorithm 12) with φ as input, generating a matrix of dimensions k * M * P , where k represents the number of classes, M the number of samples, and P the number of distinct solutions. This matrix p_s includes the predicted probabilities for each class across all samples and solutions. Subsequently, the algorithm determines the indices of maximum probabilities along the first axis, resulting in an M * P matrix $\vec{\rho}$, which indicates the predicted class for each sample under each solution. The F1 and accuracy scores for these predictions are then calculated, yielding a matrix of dimensions k * P, where each element represents the F1 score and the accuracy for a given class across all solutions. Averaging these scores across classes, the algorithm computes the macro F1 value and accuracy for each solution, resulting in a P* 1 vector f, which succinctly quantifies the model's performance across different solutions. Additionally, the algorithm updates an evaluation counter, evaNumber, to track the number of fitness evaluations.

The Employed Bee function (Algorithm 7) iteratively updates the parameters of P solutions. For each solution, parameters are randomly selected based on the MR and updated using values from a randomly chosen neighboring solution, as described in line 8. This process allows each employed bee to perform local searches within the neighborhood of its food source, thereby generating new solutions.

In Algorithm 6, probability values for each solution are determined based on their respective fitness values, such that solutions with higher fitness values are

assigned higher probability values. This approach ensures that solutions demonstrating greater efficacy are more likely to be selected for further exploration.

The Onlooker Bee function (Algorithm 8) selects solutions based on their probability values, which reflect their fitness levels. For each selected solution, parameters are randomly identified to be modified in accordance with the MR. A neighboring solution is chosen at random, and the selected parameters are updated by considering both the current and neighboring solutions' parameters. This process is repeated until all onlooker bees have been assigned a solution, allowing for local search and optimization within the solution space.

By combining the exploration capabilities of the ABC algorithm with the learning and optimization capabilities of the ANN, the proposed parallel binary and multi-class ABC-ANN algorithms aim to find optimal weight values for the neural network that result in accurate classification of network intrusions.

Algorithm 11 : The Proposed Parallel ABC-ANN multi-class classification method

Input: Input matrix X_{MxN} , target $\vec{y}_M$, number of output neurons *k*, number of food sources *P*, position of the food sources
W_{PxD}, maximum evaluation number *MEN*, lower bound *lb*, upper bound *ub*, limit, modification rate *MR*, hidden layer size *HLS*, misclassification cost matrix Ω_{kxk}
Output:

$D \leftarrow (N + 1) \times HLS + (HLS + 1) \times k$
$Y' \leftarrow OneHotEncoder(\vec{y}_M)$
$W \leftarrow CreateFoodSources(P, D)$ ▷ Algorithm 2
$W' \leftarrow W$
$\vec{f}it \leftarrow CalcFit(W)$ ▷ Algorithm 13 / 14
$\vec{\tau} \leftarrow zeros(P)$
$evaNumber \leftarrow 0$
while $evaNumber < MEN$ **do**
    SendEmployedBees() ▷ Algorithm 7
    $s\vec{f}it \leftarrow CalcFit(W')$
    $i\vec{n}d \leftarrow s\vec{f}it > \vec{f}it$
    $ri\vec{n}d \leftarrow s\vec{f}it \leq \vec{f}it$
    $\vec{\tau}\,[i\vec{n}d] \leftarrow 0$
    $W[i\vec{n}d] \leftarrow W'[\,i\vec{n}d]$
    $\vec{f}it\,[i\vec{n}d] \leftarrow s\vec{f}it\,[i\vec{n}d]$
    $\vec{\tau}\,[ri\vec{n}d] \leftarrow \vec{\tau}\,[ri\vec{n}d] + 1$
    CalcProbabilities() ▷ Algorithm 6
    SendOnlookerBees() ▷ Algorithm 8

```
    sfit ← CalcFit(W')
    for i ← 0 : P do
        t ← tmpID[i]
        if sfit [i] > fit [t] then
            τ [t] ← 0
            W[t, :] ← W'[i, :]
            fit [t] ← sfit [i]
        else
            τ [t] ←τ [t] + 1
        end if
    end for
    SendScoutBee()                                  ▷ Algorithm 9
    MemorizeBestSource()                            ▷ Algorithm 10
end while
return gpar                                         ▷ return global params
```

Algorithm 12: Calculate Output Layer (multi-class classification)

```
procedure CalcOutputLayer(φ)
    M, N ← X.shape
    P ← φ.shape[0]
    p_s ← zeros(k, M, P)                            ▷output neurons
    for i ← 0 : k do
    p_s [i, :, :] ← p_s [i, :, :] + φ[:, –k + i]    ▷ bias addition
    end for
    for i ← 0 : HLS do
        w_li ← φ[:, i × N : (i + 1) × N]^T
        b_li ← φ[:, i × N : (i + 1) × N ]^T
        z_i ← σ(X.dot(w_li) + b_li)                 ▷ σ is sigmoid func
        for j ← 0 : k do
            p_s [j, :, :] ← p_s [j, :, :] + z_i*φ[:, N × HLS + 2 × j + i]
        end for
    end for
    p_s = exp(p_s)
    t ← p_s.sum(axis = 0)                           ▷ sum p_s along the columns
    p_s ← p_s / t
    return p_s
end procedure
```

Algorithm 13: Calculate F1-score based Fitness Function (multi-class classification)

```
procedure CALC FIT (φ )
    p_s ← CALCOut putLayer(φ )
    ρ ← argmax(p_s, axis = 0)
    f 1 ← F1 score(ρ)
    f ← mean( f 1, axis = 0)
```

6: *evaNumber* ← *evaNumber* + *len(f)*
7: **return** *f*
8: **end procedure**

Algorithm 14: Calculate Accuracy Score based Fitness Function

1: **procedure** *CALC FIT (φ)*
2: p_s ← *CALCOut putLayer(φ)*
3: *ρ* ← *argmax*(p_s, *axis = 0)*
4: *acc* ← *Accuracy_score(ρ)*
5: *f* ← *mean(acc, axis = 0)*
6: *evaNumber* ← *evaNumber* + *len(f)*
7: **return** *f*
8: **end procedure**

3.3.7 Data vectorization and parallel computation on GPU

The ABC-ANN algorithm requires a robust acceleration mechanism to effectively handle big data challenges and achieve faster convergence to a global solution. To address this need, vectorization and GPU parallelization have been employed to enhance the computational efficiency of the optimization process.

Vectorization involves transforming mathematical operations into vector form, leveraging the computational capabilities of modern processors for parallel execution. By leveraging the NumPy library, which is widely used for numerical computing in Python, the code is designed to perform array operations efficiently and in parallel. The use of vectorized operations in NumPy eliminates the need for explicit looping and indexing, allowing shorter and more readable code. This method not only reduces the number of lines of code but also reduces the possibility of bugs and errors. Additionally, vectorization provides significant performance enhancements. Utilizing the underlying C implementation of NumPy enables efficient parallel execution of array operations. This provides faster execution times compared to sequential processing, where traditional loops are used. The main benefits of vectorized code are enhanced readability, decreased code complexity, and increased computational efficiency. It allows for code that is cleaner, making it simpler to understand and maintain.

Moreover, parallel execution of operations can result in significant performance improvements, particularly when dealing with large datasets or problems with high computational costs.

With the rapid advancement of GPU technologies, researchers are increasingly turning to parallel computing to boost algorithm speed. In this regard, the CuPy library for Python, developed by [70], has gained prominence. CuPy is an open-source Python library designed to harness NVIDIA GPUs for accelerating matrix operations. It is fully compatible with NumPy and allows the utilization of modern GPU capabilities through a compatible interface.

In this study, all data used in the training phase of the ABC-ANN algorithm were condensed into minimal matrices and converted into first NumPy and then Cupy arrays to optimize calculation speed. Overall, the utilization of vectorization and GPU parallelization via the CuPy library in the ABC optimization code of this study enhances the efficiency and readability of the implementation, rendering it a valuable tool for scientific computing and optimization tasks.

3.3.8 Bayesian optimization

Bayesian optimization is a technique that leverages Bayes' theorem to efficiently search for the global optimum value of an objective function. It involves constructing a probabilistic model, known as the surrogate function, which represents the objective function. This surrogate function is then iteratively evaluated and updated based on the observed results.

In the context of ML, Bayesian optimization is commonly used for hyperparameter tuning process. Hyperparameters are configuration settings of a model that are not acquired from the data but need to be specified by the user. Identifying the optimal set of hyperparameters is essential for maximizing the performance of a machine learning model on a specific dataset.

Hyperparameter tuning is a challenging task as it involves searching through a large space of possible hyperparameter values. The objective function, usually representing the model's performance on a validation set, is often complex and computationally expensive to evaluate.

Bayesian optimization provides a systematic approach to efficiently searching for the optimal hyperparameters. Unlike random or grid search methods, Bayesian optimization maintains a record of previous evaluation results.

These results are used to build a probabilistic model that correlates hyperparameters with the probability of obtaining a specific score on the objective function. It constructs a probabilistic model of the objective function based on the

observed evaluations and uses this model to guide the search process. By iteratively selecting promising hyperparameter configurations based on an acquisition function, Bayesian optimization gradually explores the hyperparameter space and converges towards the optimal solution. Bayesian optimization can explore a larger search space compared to more traditional hyperparameter optimization techniques like grid search and random search, thereby achieving more effective results in relatively shorter time frames.

In this work, for this purpose, one of the Python libraries called Hyperopt [71] is used. Hyperopt is a library for Bayesian optimization that can implement the Tree-structured Parzen Estimator (TPE), which is selected for this study and is more advanced than other optimization algorithms. There are four components of Bayesian optimization:

1. Objective function (F(x)): The function we aim to minimize.
2. Domain space (X): The range of parameter values over which the objective is minimized.
3. Hyperparameter optimization function (TPE): This function creates the surrogate function and selects the next values to assess.
4. Trials: Each instance where we evaluate the objective function, recording the score and parameter pairs.
5. Max_eval: maximum evaluation number.

TPE is a bayesian-based approach that tries to build a probabilistic model. TPE implies that hyperparameter space exhibits a tree-like structure: the selection of a value for one hyperparameter determines the subsequent selection of another hyperparameter and the range of values available for it. The TPE algorithm works like below:

1. Create a randomly chosen initial point from domain space X: x*.
2. Compute F(x*). (The function F corresponds to the objective function, which in our case is the negative of accuracy.)
3. Utilizing the trial history, construct the conditional probability model P(F | x).
4. Select based on P(F | x), anticipating an improvement in F(x*).
5. Calculate the actual value of F(x*).
6. Iterate through steps 3-5 until one of the stopping criteria is met, such as i>max_eval. (The goal is to find the global minimum of F(x) over X.)

Table 3.4 displays the hyperparameter ranges for various algorithms, encompassing the newly proposed ABC_ANN method alongside other comparative techniques. Table 3.5 presents the optimal parameters achieved after 150 iterations using the Bayesian optimization algorithm. In order to demonstrate the effectiveness of the Bayesian optimization algorithm, its performances have been compared with the randomized search algorithm. It can be seen in Table 3.6 that although the randomized search algorithm was run with 250 iterations, it could not pass the Bayesian optimization algorithm in terms of evaluation metrics including f1-score, accuracy, DR, and FAR.

Table 3. 4 Hyperparameter ranges of Bayesian optimization based on different classification algorithms.

Model	Parameters	Range
DAE	learning rate	[10e-8,10e-1]
	hidden size 1	[100,150] and [25,35]
	hidden size 2	[30,100] and [10,25]
	dropout rate 1	[0,0.3]
	dropout rate 2	[0,0.3]
	batch size	[1,1024]
	epochs	[1,100]
	act. func.	tanh, sigmoid
PSO_ANN	no. of particles	[3,20]
	c1	[0.5,3]
	c2	[0.5,3]
	no. of iter.	[5,100]
	w	[0.1,2]
	hidden size	[5,100]
	act. func.	tanh, sigmoid, relu
GA_ANN	no. of solutions	[10,20]
	no. of generations	[10,100]
	hidden size	[3,20]
	no. of parents mating	[1,10]
	act. func.	tanh, sigmoid, relu
SGD_ANN	batch size	8,16,32,64,128,256
	epoch	[50,200]
	hidden size	[3,50]
	dropout rate	[0,0.4]
	learning rate	[0.01,0.5]
	momentum	[0.1,1]

	act. func.	tanh, sigmoid, relu
Adam_ANN	batch size	8,16,32,64,128,256
	epoch	[50,200]
	hidden size	[3,50]
	dropout rate	[0,0.4]
	learning rate	[0.01,0.5]
	act. func.	tanh, sigmoid, relu
Proposed ABC_ANN	HLS	[2,20]
	lb	[-30,0]
	ub	[0,30]
	evaluation number	[10000,120000]
	limit	[10,200]
	P	[10,200]
	MR	[0.01,0.2]
	threshold	[0.2,0.8]
	act. func.	tanh, sigmoid

3.4 Results and Discussion

3.4.1 Datasets and data preprocessing

This study utilizes the UNSW-NB15 and the NF-UNSW-NB15-v2 datasets to build the NIDS model. The UNSW-NB15 dataset consists of a combination of actual modern normal network activities and synthesized current network attack activities. It serves as an alternative to older benchmark datasets and is widely adopted for evaluating the performance of Network Intrusion Detection Systems (NIDS).

The NetFlow-based format of the UNSW-NB15 dataset, referred to as the NF-UNSW-NB15-v2 [72], has been expanded with supplementary NetFlow attributes and labeled with corresponding attack categories. The dataset comprises a total of 43 features and 2,390,275 data flows, with 95,053 classified as attack samples and 2,295,222 as benign. As part of the data pre-processing to avoid and mitigate bias in model training, six attributes are removed from the dataset. These include minimum or maximum traffic TTLs, port numbers, IPv4 source, and destination addresses that do not significantly contribute to the classification performance and are highly correlated with class labels.

Since this dataset does not provide users with pre-existing training and test sets, the experiments split the data into partitions containing an equal proportion of

samples from both the benign and attack classes. Specifically, 33% of the data is allocated for testing, while the remaining portion constitutes the training set.

The training set of the UNSW-NB15 dataset contains 175,341 samples with 45 features. Among these samples, 56,000 are labeled as "normal" traffic, while 119,341 samples are labeled as "abnormal" traffic, representing various types of attacks. The testing set consists of 82,332 samples with the same feature size as the training set. Within the testing set, 37,000 samples are categorized as "normal" traffic, and the remaining 45,332 samples represent "abnormal" traffic with different attack types, including DoS, backdoors, generic attacks, analysis attacks, exploits, shell code, fuzzers, reconnaissance, and worms. So, within the UNSW-NB15 dataset containing categorical features, a category encoder technique is utilized to convert these categories into numerical values. Specifically, "service," " proto," and "state" are the categorical features in the dataset. Following the encoding process, the total number of features expands from 45 to 197.

To minimize the impact of different scales across features and reduce computational costs and training time, normalization techniques are applied to both datasets. Several normalization strategies exist in the literature, such as the Max-Abs scaler, the Standard scaler, and the Min-Max scaler. Considering the sparsity analysis of both datasets, which indicates a significant proportion of zeros, the Max-abs scaler is appropriate for these datasets.

The Max-abs normalization technique scales each feature by its maximum absolute value while preserving all zeros. This normalization approach is used to scale all values in the dataset to the range of [0, 1]. By applying the Max-abs scaler, datasets are prepared for the following processing and analysis in the NIDS model construction.

Table 3. 5 The optimal parameters found by the Bayesian hyperparameter optimization algorithm on the NF-UNSW-NB15-v2 and UNSW-NB15 datasets.

Model	Parameters	Opt. Values UNSW-N15	Opt. Values NF-UNSW-N15-v2
DAE	learning rate	0.3	0.24
	hidden size 1	100	31
	hidden size 2	53	25
	dropout rate 1	0.1	0.1

	dropout rate 2	0.0	0.1
	batch size	256	37
	epochs	90	12
	act. func.	sigmoid	sigmoid
PSO_ANN	no. of particles	11	8
	c1	1.36	1.25
	c2	1.88	2.3
	w	0.38	0.48
	hidden size	3	3
	act. func.	sigmoid	sigmoid
	no. of iter	49	37
GA_ANN	no. of solutions	14	7
	no. of generations	24	10
	hidden size	3	3
	no. of parents mating	8	4
	act. func.	tanh	tanh
SGD_ANN	batch size	16	128
	epoch	108	190
	hidden size	35	42
	dropout rate	0.1	0.3
	learning rate	0.08	0.003
	momentum	0.35	0.02
	act. func.	relu	tanh
Adam_ANN	batch size	32	128
	epoch	88	113
	hidden size	25	15
	dropout rate	0	0.1
	learning rate	0.17	0.058
	act. func.	sigmoid	relu
Proposed ABC_ANN	HLS	3	4
	lb	-20	-14.6
	ub	20	13.8
	evaluation number	60008	58567
	limit	50	69
	P	40	68
	MR	0.054	0.04
	threshold	0.5	0.5
	act. func.	sigmoid	sigmoid

3.4.2 Evaluation metrics

Evaluation metrics are essential for assessing the performance of machine learning algorithms. In addition to accuracy, it is important to take into account the F1 score, the FAR, and the DR to gain a comprehensive understanding of the model's

performance. A confusion matrix (Table 2.2) provides a detailed breakdown of correct and incorrect predictions made by the model across different classes [73]. Evaluation metrics and confusion matrix play crucial roles in assessing the performance of ML models, providing insights into their predictive capabilities, and helping in the refinement and optimization of models for better performance. Some common evaluation metrics derived from the confusion matrix include precision, recall, accuracy, F1 score, DR, and FAR.

The definition of accuracy is the ratio of correctly estimated samples to all samples (Equation (2.1)). It provides a basic performance measure but may not be sufficient, especially in the case of unbalanced datasets. As is often the case with problems with network intrusion detection, imbalance occurs when there is an insufficient number from one class (e.g., abnormal). In such cases, the importance of metrics such as the F1 score becomes even more apparent. The F1 score is a statistical measure of precision and recall (Equation (3.5)). Precision measures the ratio of true positives (correctly predicted abnormal samples) to the total number of predicted positives, while recall measures the ratio of actual positives to the overall actual positive number. The F1 score is the harmonic means of precision and recall, providing a balanced measure of accuracy, taking into account both false positives and false negatives.

DR, also known as TPR or Sensitivity, is the ratio of true positive samples to the total number of actual positive samples (abnormal samples in the dataset) (Equation (2.2)). It indicates the ability of the model to correctly identify positive instances.

FAR, also known as False Positive Rate (FPR), is the ratio of false positive samples to the total number of actual negative samples (normal samples in the dataset) (Equation (2.3)). It represents the proportion of normal instances that are incorrectly classified as abnormal.

By considering these evaluation metrics, including F1-score, DR, and FAR, alongside accuracy, a more comprehensive assessment of the binary classification model's performance can be obtained, particularly in the presence of imbalanced datasets. In such cases, the focus is not only on overall accuracy but also on correctly identifying abnormal instances while minimizing false alarms.

Some metrics are specifically designed for different ML tasks, such as multi-class classification. In the context of multi-class classification, various metrics are employed during the performance evaluation stage. These metrics are derived from the

confusion matrix (Table 2.2) and include precision-weighted, precision-macro, recall-weighted, recall-macro, f1-macro (calculated as the arithmetic mean of each per-class f1 score) (Equation (3.6)), f1-weighted (computed as the mean of all per-class f1 scores, taking into account each class's support) (Equation (3.7)), and accuracy.

These metrics are particularly important for multi-class classification problems, as they enable the generation of a reliable final model by considering the performance across multiple classes. By utilizing these evaluation metrics, the model's precision, recall, f1 scores, and overall accuracy can be determined, providing valuable insights into its effectiveness. (In Equation (3.6) and Equation (3.7), n = no. of classes), (In Equation (3.7), w_i = no. of samples in class i / total no. of samples).

$$\text{F1 score} = \frac{TP}{TP+\frac{1}{2}(FP+FN)} \tag{3.5}$$

$$\text{F1 – macro} = \frac{1}{n}\sum_{i=0}^{n}(\text{F1 score})_i \tag{3.6}$$

$$\text{F1 – weighted} = \sum_{i=0}^{n}(w)_i * (\text{F1 score})_i \tag{3.7}$$

3.4.3 Experimental setup

The proposed methods were carried out utilizing the Colab platform offered by Google, which provides access to NVIDIA T4 GPUs. The GA, PSO, and ANN with SGD and Adam optimization algorithms were conducted using PyGAD [74], Pyswarms [75], and Tensorflow [76] libraries, respectively.

3.4.4 Experimental results

The experimental setup of this study encompasses four primary processes. Firstly, the objective is to evaluate the contribution of newly extracted features obtained via DAE to the binary classification task. These features were derived utilizing a DL technique, and their effect on the classification outcomes was assessed.

Secondly, an exploration into the impact of the number of selected features on the classification performance is conducted using XGBoost with 5-fold cross-validation, which ensures the selection of the most appropriate features. This investigation entails applying the proposed DAE-based ABC-ANN algorithm with default hyperparameter settings to three distinct feature sets:

- Original features

- Encoded features obtained from the DAE
- Concatenation of the original and encoded features

To ascertain the optimal number of features, we employ a 5-fold cross-validation approach, augmented by XGBoost feature selection. The objective is to systematically vary the number of selected features and evaluate their impact on classification metrics, including accuracy, F1-score, DR, and FAR.

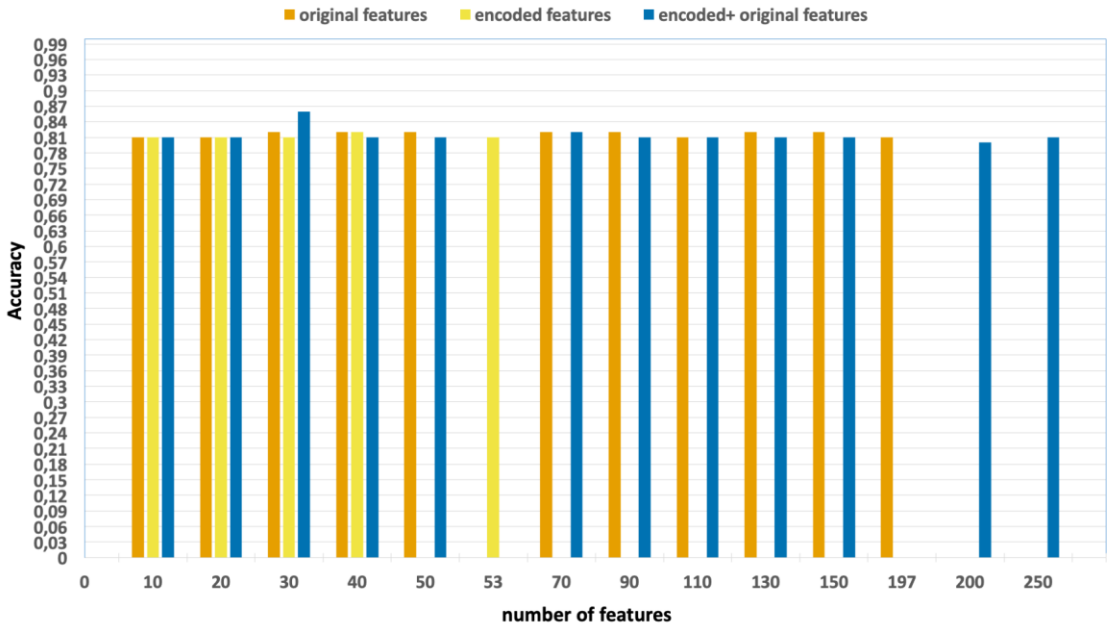

Figure 3. 3 Accuracy of the UNSW-NB15 dataset values according to different number of feature subsets obtained from the 5foldcv XGBoost algorithm.

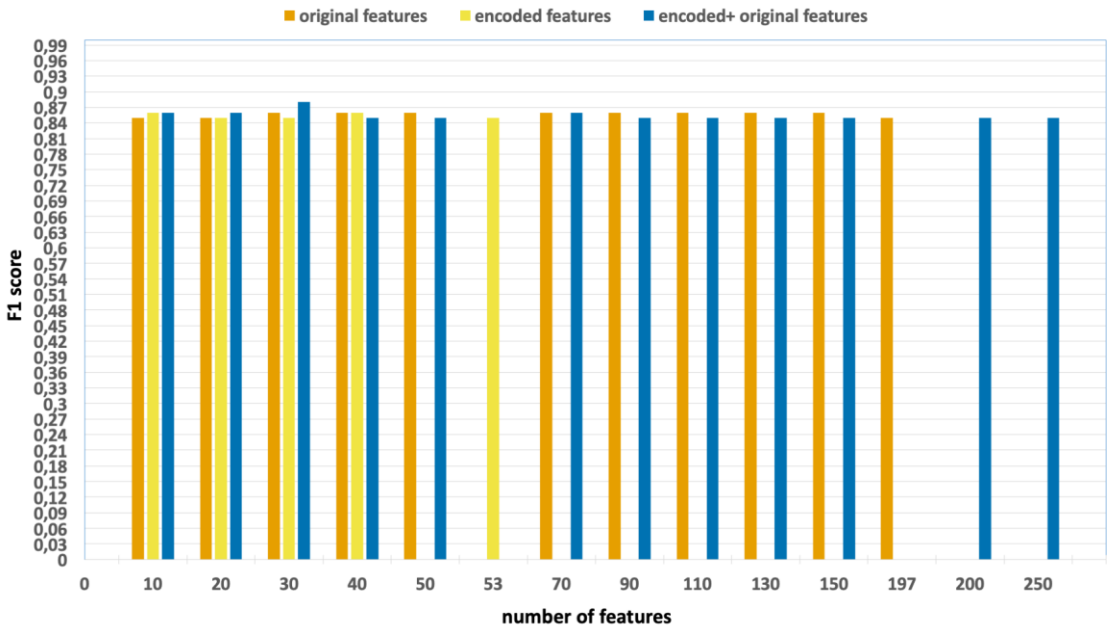

Figure 3. 4 F1 scores of the UNSW-NB15 dataset according to different number of feature subsets obtained from the 5foldcv XGBoost algorithm.

During this procedure, a total of 30 features are selected from the UNSW-NB15 dataset, and 40 features are selected from the NF-UNSW-NB15-v2 dataset based on the 5-fold cross-validation XGBoost feature selection technique, as detailed in Section 3.3.2.

Ablation studies were conducted with the aim of evaluating the individual contributions of each process, including feature extraction, feature selection, and the effects of parallelization and vectorization on training times. Figure 3.3 and Figure 3.4 present the accuracy and F1 results derived from various configurations for the UNSW-NB15 dataset. These configurations include:

- Results obtained without feature extraction and selection, which exhibit accuracy and F1 scores of 0.81 and 0.86, respectively.
- Results obtained solely with extracted encoded features, yielding accuracy and F1 scores of 0.81 and 0.85, respectively.
- All accuracy and F1 scores obtained with different feature sets.

Similarly, Figure 3.5 and Figure 3.6 encompass the accuracy and F1 results derived from various configurations for the NF-UNSW-NB15-v2 dataset. These configurations include:

- Results obtained without feature extraction and selection, which exhibit accuracy and F1 scores of 0.98 and 0.79, respectively.
- Results obtained solely with extracted encoded features, yielding accuracy and F1 scores of 0.99 and 0.81, respectively.
- All accuracy and F1 scores obtained with different feature sets.

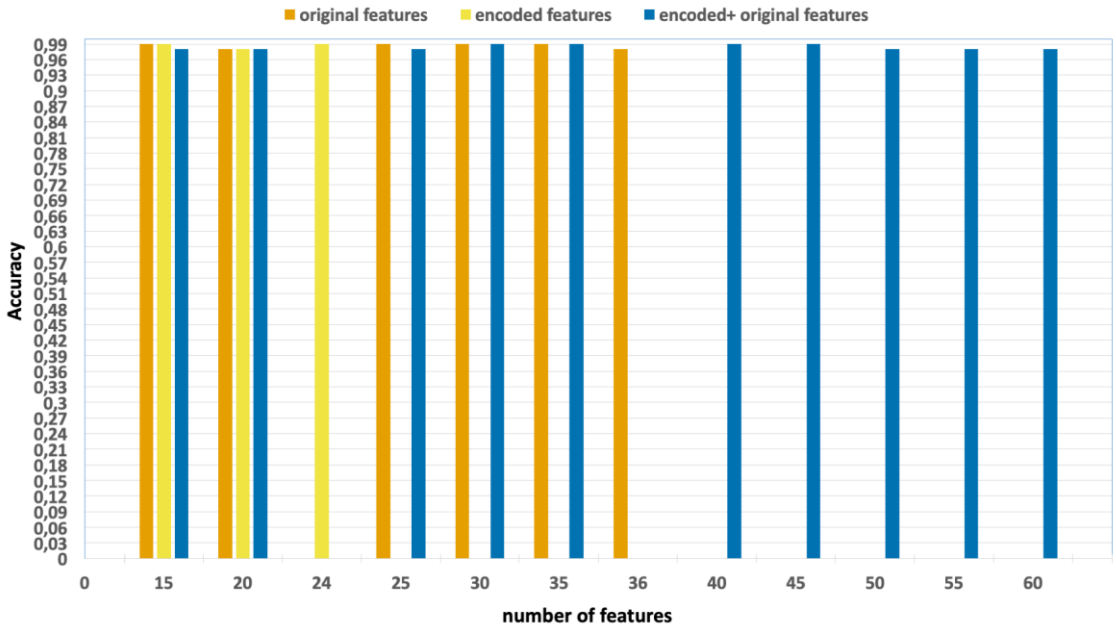

Figure 3. 5 Accuracy of the NF_UNSW-NB15_v2 dataset values according to different number of feature subsets obtained from the 5foldcv XGBoost algorithm.

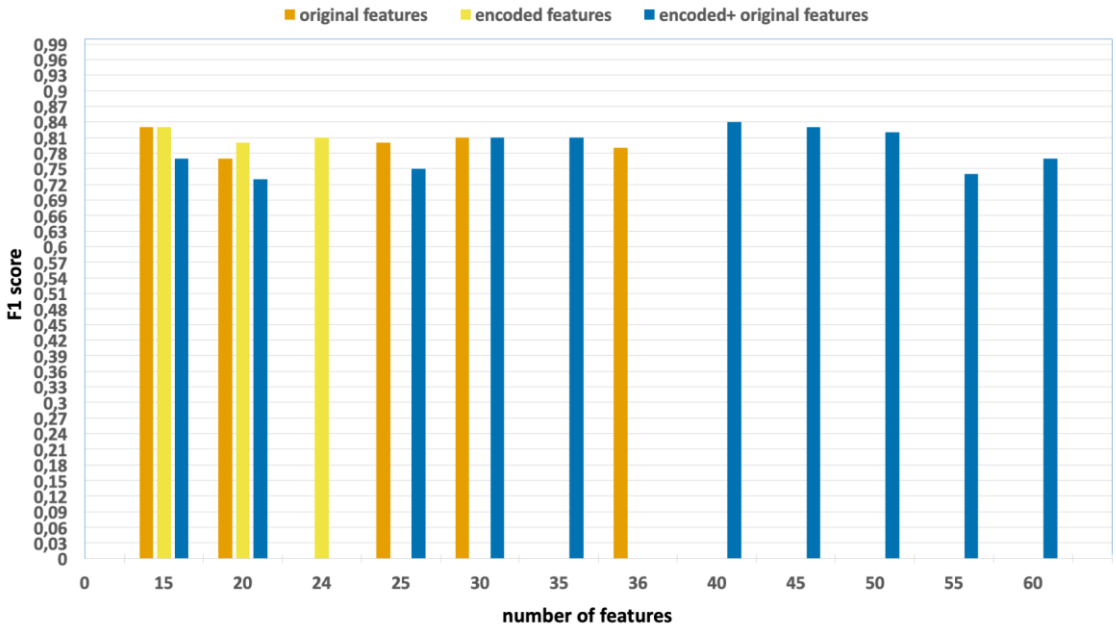

Figure 3. 6 F1 scores of NF-UNSW-NB15-v2 dataset according to different number of feature subsets obtained from the 5foldcv XGBoost algorithm.

Thirdly, the performance of the proposed DAE-based ABC-ANN method has been compared with benchmark metaheuristics, namely the GA and Particle Swarm Algorithm (PSO). Furthermore, it has been compared to the performance of the

proposed DAE-based ABC-ANN method with a conventional ANN approach that involves error back propagation and weight adjustment using a Stochastic Gradient Descent (SGD) and Adam optimization algorithms. In this study, with the goal of minimizing computational load, all matrices referenced in Algorithm 1 were vectorized utilizing the Python programming language and numpy library, as opposed to Python lists and loops (a detailed explanation can be found in Section 3.3.7). This represents a significant contribution to literature. Thus, the computational load is substantially reduced to a minimum. In consideration of the big datasets as in this study, it became crucial to accelerate the model. In order to accomplish this, it has also utilized GPU parallelization for vectorized loops, which significantly increased the overall speed and efficiency. The utilization of CuPy, an open-source library specifically developed for accelerating matrix operations on NVIDIA GPUs, enabled this acceleration.

Table 3. 6 The best performance evaluation results of the UNSW-NB15 dataset with 30 selected features and the NF-UNSW-NB15-v2 dataset with 20 selected features, calculated using the Bayesian hyperparameter optimization algorithm with 150 iterations and the randomized search strategy with 250 iterations.

Dataset	Model	Opt. Strategy	No. of iter.	Acc	F1	DR	FAR	TTime
UNSW-NB15	proposed DAE-ABC-ANN (GPU)	Randomized Search	250	0.82	0.86	0.75	0.008	6min 16sec
		Bayesian Opt.	**150**	**0.86**	**0.88**	**0.81**	**0.005**	**3min 23sec**
NF-UNSW-NB15-v2	proposed DAE-ABC-ANN (GPU)	Randomized Search	250	0.99	0.84	0.73	0.0008	9min 58sec
		Bayesian Opt.	**150**	**0.99**	**0.89**	**0.81**	**0.0003**	**8min 41sec**

These all-aforementioned classification algorithms were applied to the optimal set of 30 and 40 features selected in the previous step using XGBoost feature selection. After Bayesian optimization has been conducted on all classification algorithms with 150 iterations, the optimum results of this comparison are presented in Table 3.7 and 3.8.

Table 3. 7 The best performance evaluation results of the UNSW-NB15 dataset with 30 selected features, calculated using the Bayesian hyperparameter optimization algorithm after 150 iterations.

Model	Accuracy	F1	DR	FPR	Training Time
GA ANN	0.75	0.81	0.72	0.15	54min 23sec
PSO ANN	0.81	0.85	0.74	0.006	34min 31sec
SGD ANN	0.82	0.86	0.76	0.016	7min 25sec
Adam ANN	0.84	0.87	0.79	0.032	4min 14sec
proposed DAE-ABC-ANN (CPU)	0.86	0.88	0.81	0.005	23min 47sec
proposed DAE-ABC-ANN (GPU)	**0.86**	**0.88**	**0.81**	**0.005**	**3min 23sec**

Table 3. 8 The best performance evaluation results of the NF-UNSW-NB15-v2 dataset with 40 selected features, calculated using the Bayesian hyperparameter optimization algorithm after 150 iterations.

Model	Accuracy	F1	DR	FPR	Training Time
GA ANN					2hr 38min
PSO ANN	0.82	0.49			4hr 40min
SGD ANN	0.99	0.86	**0.875**	0.006	50min 42sec
Adam ANN	0.99	0.87	0.84	0.003	9min 28sec
proposed DAE-ABC-ANN (CPU)	0.99	0.89	0.81	0.0003	2hr 16min
proposed DAE-ABC-ANN (GPU)	**0.99**	**0.89**	0.81	**0.0003**	**8min 41sec**

Furthermore, for the purpose of ensuring the results' reliability, by executing the models 20 times for the NF-UNSW-NB15-v2 and UNSW-NB15 datasets, the best, worst, average training time, and standard deviation values were recorded for each classifier. Table 3.9 summarizes the best, worst, average, and standard deviation values obtained after repeating the best model achieved in the UNSW-NB15 dataset 20 times. Table 3.10, on the other hand, provides a summary of the same results obtained for the NF-UNSW-NB15-v2 dataset. Overall, the experimental setup has involved evaluating

the contribution of DAE-extracted features, exploring the influence of the number of selected features, comparing the performance of the proposed hybrid DAE-based ABC-ANN method with benchmark metaheuristics, and contrasting it with conventional ANN approaches with SGD and Adam optimization using the sklearn TensorFlow library.

The results demonstrate that the proposed hybrid DAE-based ABC-ANN approach outperforms state-of-the-art algorithms in terms of accuracy, F1 score, DR, false positive rate (FPR), and training time on the NF-UNSW-NB15 and UNSW-NB15 datasets.

Table 3. 9 The time in seconds required to train each classifier on the UNSW-NB15 dataset

Model	Best time	Worst time	Avg. time	Std.
GA ANN	5454	6169	5,698.68	251.96
PSO ANN	4326	4499	4,407.37	51.78
SGD ANN	272	791	431.63	142.05
Adam ANN	209	552	316.63	79.77
proposed DAE-ABC-ANN (CPU)	1412	1533	1,442.42	34.32
proposed DAE-ABC-ANN (GPU)	**144**	**176**	**146.68**	**7.14**

Table 3. 10 The time in seconds required to train each classifier on the NF-UNSW-NB15-v2 dataset.

Model	Best time	Worst time	Avg. time	Std.
GA ANN	8760	9507	9,080.8	339.386
PSO ANN	16816	17625	1,7206	433.615
SGD ANN	2951	4394	4,519.33	544.72
Adam ANN	519	857	636.25	104.549
proposed DAE-ABC-ANN (CPU)	8040	8160	8,086	49.98
proposed DAE-ABC-ANN (GPU)	**504**	**509**	**506.94**	**1.545**

Lastly, the proposed multi-class ABC-ANN classification model was tested on the UNSW-NB15 dataset. Given the imbalanced nature of the dataset, it is crucial to focus on achieving high metrics not only for weighted accuracy, precision, and recall but also for their macro counterparts. Improving the macro metrics is important because it not only enhances the accuracy of the classes with a higher number of records but also significantly improves the accuracy of classes with a lower number of records. This approach allows for more effective monitoring of all classes.

The macro parameters derived from the multi-class classification indicate that certain attack classes with fewer records, such as DoS, Reconnaissance, Analysis, Backdoors, Shellcode, and Worms, are exhibiting signs of underfitting. To address the dimensionality problem, a new dataset is generated by concatenating the records from these six categories of attacks. This approach aims to mitigate the issues related to the limited representation of these attack classes, ultimately improving the performance of the model in detecting them.

Table 3.11 and Table 3.12 present a comprehensive summary of the performance results obtained from the utilization of the proposed parallel multi-class ABC-ANN method, employing the f1 macro-based fitness function. The outcomes are compared with other metaheuristics combined with ANNs. To derive the ANN GA results, the PyGAD open-source Python 3 library is employed. Meanwhile, the implementation of other ML approaches relies on the scikit-learn library [77]. The proposed parallel ABC-ANN method is developed using the Python programming language [78].

The proposed ABC-ANN model is structured to incorporate a hidden layer with three neurons. The activation function for this hidden layer is specified as "sigmoid." The additional parameters for the GA are set with "num_solutions" as 10, "num_generations" as 10, and "num_parents_mating" as 5. The parameters for the proposed parallel ABC-ANN are configured for network intrusion detection tasks with a lower bound (lb) set at -25, an upper bound (ub) set at 15, an evaluation number set to 100000, a limit set at 150, the number of food sources (P) set to 20, and a modification rate (MR) set at 0.02.

The results strongly establish the superiority of the proposed multi-class parallel ABC-ANN method over some alternative approaches across all evaluation metrics, as indicated in Section 3.4.2. The proposed model demonstrates the highest F1M score (0.64) among all methods, showcasing superior performance. ANN GA has

the longest training time (1666 seconds) and relatively lower performance metrics compared to the proposed multi-class parallel ABC-ANN. While SGD and Adam demonstrate shorter training times than the proposed multi-class model, it is evident that the proposed approach emerges as a superior option when the balance between classification performance and execution times is taken into account. Moreover, Table 3.12 vividly demonstrates the remarkable impact of GPU parallelization on the proposed model, leading to a substantial acceleration in speed. In fact, compared to CPU execution, GPU parallelization achieves an impressive 17-fold increase in processing speed.

Table 3. 11 Performance results and training times of the proposed ABC-ANN and combination of other metaheuristics and ANN with default parameters on the UNSW-NB15 dataset.

Method	F1M	F1W	PreM	PreW	RecM	RecW	Acc	ttime
ANN_ Adam	0.50	0.65	0.56	0.72	0.54	0.63	0.63	16sec
ANN_ SGD	0.59	0.71	0.62	0.79	0.62	0.67	0.67	308sec
ANN_GA	0.49	0.62	0.52	0.69	0.53	0.61	0.61	1666sec
Proposed parallel multi-class ABC-ANN	**0.64**	**0.72**	**0.66**	**0.82**	**0.69**	**0.68**	**0.68**	**998sec**

F1M=F1Macro, F1W= F1Weighted, PreM= Precision macro, PreW= Precision weighted, RecM= Recall macro, RecW= Recall weighted, Acc= Accuracy, ttime= training time.

Table 3. 12 CPU and GPU performance results and the training times of the proposed method on the UNSW-NB15 dataset

Method	F1M	F1W	PreM	PreW	RecM	RecW	Acc	ttime
Proposed parallel multi-class ABC-ANN on **CPU**	0.64	0.72	0.66	0.82	0.69	0.68	0.68	286min 59sec
Proposed parallel multi-class ABC-ANN on **GPU**	0.64	0.72	0.66	0.82	0.69	0.68	0.68	16min 38sec

F1M=F1 Macro, F1W= F1 Weighted, PreM= Precision macro, PreW= Precision weighted, RecM= Recall macro, RecW= Recall weighted, Acc= Accuracy, ttime= training time.

Chapter 4

Conclusions and Future Prospects

4.1 Conclusions

This thesis makes significant contributions to the field of network intrusion detection through three distinct proposed studies.

Firstly, this thesis mainly aims to fill a gap in the literature by evaluating wired, wireless, and SDN networks from different perspectives using various state-of-the-art and hybrid ML strategies to develop efficient network intrusion detection systems. The focus is on addressing class imbalance problems, performing feature selection and extraction, and conducting binary classification tasks effectively. To achieve this, this study utilizes the SMOTE, Adasyn, and TomekLink algorithms for handling class imbalances on wired, wireless, and SDN networking datasets. The XGBoost feature selection methodology has been employed to identify the most informative features. For the binary classification task, several ML methods have been applied, including SVM, KNN, XGBoost, Random Forest, and AE-based ensemble classifiers. The datasets used in this study are publicly accessible wireless (AWID), wired (UNSW-NB15), and SDN (InSDN) network intrusion datasets. The performance of the proposed intrusion detection systems in this study has been assessed across important conditions, such as with or without feature selection and using imbalanced or balanced datasets. When optimized on the validation set and evaluated on the test set, these developed models for wired, wireless, and SDN networks perform well in terms of binary classification. Balanced versions of the datasets perform better than their imbalanced counterparts. Table 2.7 provides a summary of the best performance results for each dataset in terms of F1-measure, overall accuracy, DR, and FAR. In addition, Table 2.1 demonstrates that the combination of the ML methodologies proposed in this study generates superior outcomes in comparison to the existing literature.

Importantly, the study demonstrates that reliable NIDS can be generated with oversampling techniques, efficient feature selection techniques, and cost-effective tree-based algorithms. Additionally, it is noteworthy that the proposed intrusion detection systems achieve significant performance that is close to when used 20 features, while using only 12 features on the AWID dataset. Accuracy values are almost equal on both feature subsets (Table 2.4).

Overall, this thesis makes a valuable contribution to the literature in the field of network intrusion detection by offering valuable insights into effective strategies for handling class imbalance, feature selection, and binary classification tasks in wired, wireless, and SDN networks. As far as we know, no study has been undertaken to evaluate different network intrusion datasets, such as wired, wireless, and SDN, together, considering class imbalance, feature selection, and hyperparameter optimization tasks. The performance results highlight the success of the proposed methods and their potential for practical implementation in real-world network security scenarios. The optimal results, considering F1 measure, overall accuracy, DR, and FAR, have been achieved for the UNSW-NB15, preprocessed AWID, and InSDN datasets, with values of [0.9356, 0.9289, 0.9328, 0.07597], [0.997, 0.9995, 0.9999, 0.0171], and [0.9998, 0.9996, 0.9998, 0.0012], respectively. The outcome demonstrates the model's potential capacity for detecting intrusions.

Secondly, this study combines DAE with vectorized and GPU-parallelized ABC-ANN to efficiently address big data problems by searching for global solutions in a faster manner. While existing methods may achieve high accuracy, they may suffer from high training times, low DR, and computational complexity. In this study, the ABC algorithm has been vectorized and coded to run in parallel on GPUs to address these issues. Additionally, DAE and feature selection have been conducted to obtain a more robust dataset representation.

The proposed DAE-based ABC-ANN method is compared with the conventional ANN back propagation (ANN-BP), ANN-PSO, ANN-GA, and ANN-Adam optimization algorithms, and the results are thoroughly analyzed. The ABC algorithm in the ANN training phase allows for the avoidance of local minimum solutions by conducting a high-performance search in the solution space.

This study investigated the XGBoost algorithm for feature selection and the DAE for feature extraction in conjunction with numerous approaches, including PSO, GA, SGD, and Adam optimization, to develop reliable, efficient, and accurate IDSs. In order to evaluate the effectiveness of these techniques, the benchmark UNSW-NB15 and up-to-date NF-UNSW-NB15-v2 datasets were trained and tested. First, the DAE-based feature extraction method was conducted with bayesian hyperparameter optimization on datasets in order to extract the most representative features, and it resulted in 53 encoded features on the UNSW-NB15 and 24 encoded features on the NF-UNSW-NB15-v2 datasets. Second, the XGBoost-based feature selection method was used to select the best features from the combination of original and encoded features. Third, an ABC-ANN is proposed with CPU and GPU parallelization, which allows the use of ABC intelligence in big data problems. The computational costs of the proposed ANN-ABC method impose limitations on the GPU. Last, a Bayesian-based hyperparameter optimization technique is conducted on all experimental algorithms and the proposed ABC-ANN algorithm in order to find the best hyperparameter combinations that improve detection accuracy and f1 score.

To place our findings in context, this study has conducted a comprehensive literature analysis. In addition, it has created a summary of the performance results acquired by the various algorithms and compared them using the proposed method. Consequently, the results have demonstrated clearly that the proposed DAE-based ABC-ANN method is superior to alternative approaches across all evaluation metrics mentioned in Section 3.4.2. The experimental results reveal a notable improvement in network intrusion detection through this proposed approach, exhibiting an increase in DR by 0.76 to 0.81 and a reduction in FAR by 0.0016 to 0.005 compared to the ANN-BP algorithm on the UNSW-NB15 dataset (Table 3.7). Furthermore, there is a reduction in FAR by 0.006 to 0.0003 compared to the ANN-BP algorithm on the NF-UNSW-NB15-v2 dataset (Table 3.8). These findings underscore the effectiveness of our proposed approach in enhancing network security against network intrusions.

Thirdly, this thesis introduces the parallel multi-class ABC-ANN method that capitalizes on the strengths of the ABC algorithm for optimization and the predictive capabilities of ANN for handling nonlinear patterns. To overcome the computational challenges associated with training the ABC-ANN algorithm on large datasets, the study

leverages vectorization and CPU/GPU parallelization techniques. These strategies significantly reduce the training time compared to traditional approaches.

To assess the effectiveness of the proposed multi-class ABC-ANN, comprehensive comparisons are conducted with state-of-the-art techniques. Multiple metaheuristics combined with ANN and various ML algorithms are trained and evaluated on the same datasets. Performance metrics, including f1 macro, accuracy, and training time, are considered as the evaluation criteria.

The comparative performance evaluations reveal that the proposed multi-class ABC-ANN model outperforms several commonly used state-of-the-art methods in terms of f1 macro, accuracy, and notably, training time. These results emphasize the efficiency and effectiveness of the parallel multi-class ABC-ANN method in tackling complex problems such as intrusion detection.

The following is a summary of the principal findings of our third study:

- This study proposes the parallel multi-class ABC-ANN method: Capitalize on the strengths of the ABC algorithm for optimization and the predictive capabilities of ANN for handling nonlinear patterns. The parallel multi-class ABC-ANN model proposes f1 macro as a fitness function and focuses on intrusion detection and attack classification. It utilizes the well-known UNSW-NB15 benchmark intrusion dataset and achieves improved performance compared to state-of-the-art techniques.
- This study addresses computational challenges through the utilization of vectorization and CPU/GPU parallelization techniques, resulting in a significantly reduced training time compared to traditional approaches. GPU parallelization, in particular, facilitated a remarkable 17-fold increase in processing speed.
- Comprehensive performance comparisons and evaluations were conducted by comparing the proposed parallel multi-class ABC-ANN model with multiple metaheuristics combined with ANN and various ML algorithms. The evaluation used performance metrics such as f1 macro, accuracy, and training time.
- The superiority of the proposed parallel multi-class ABC-ANN method is evident, as it outperformed alternative approaches across all evaluation metrics and achieved remarkable speed acceleration with GPU parallelization.

As a result, by leveraging the proposed optimization technique, our approach outperforms conventional methods in terms of computational efficiency and classification metrics. The best performance, as measured by F1-macro, F1-weighted, and overall accuracy metrics, was observed in the UNSW-NB15 dataset, yielding values of [0.64, 0.72, 0.68].

4.2 Societal Impact and Contribution to Global Sustainability

The research presented in this thesis holds significant implications for societal impact and contributes to global sustainability by addressing critical challenges in cybersecurity and network intrusion detection. The United Nations Sustainable Development Goals (SDGs) are a set of 17 interconnected global objectives adopted by all United Nations Member States in 2015. Spanning a wide range of issues such as economic growth, education, sustainable cities, innovation, and infrastructure, the SDGs aim to address the world's most pressing challenges in a comprehensive and integrated manner. In this context, this thesis aligns with SDGs including Decent Work and Economic Growth (SDG 8), Peace, Justice, and Strong Institutions (SDG 16), Sustainable Cities and Communities (SDG 11). and Innovation and Infrastructure (SDG 9).

The escalating complexity of cyberattacks poses a substantial threat to digital infrastructure worldwide, as emphasized by SDG 9. By employing Artificial Neural Networks (ANNs) and proposing a novel Deep Autoencoder (DAE)-based Parallel Artificial Bee Colony (ABC) algorithm for network intrusion detection, this research offers a promising way to support network security. The proposed approach demonstrates notable improvements in detection rates and false alarm rates, thereby strengthening defenses against malicious intrusions.

The proliferation of computer networks underscores the importance of robust intrusion detection systems (IDSs) in safeguarding sensitive information from unauthorized access and cyber threats, as emphasized by SGD 11. By comprehensively assessing class imbalances and classification performance across wired, wireless, and Software-Defined Networking (SDN) environments, this study contributes to the optimization of IDS efficiency. Through machine learning techniques such as

oversampling, feature selection, and Bayesian optimization, the research achieves significant advancements in classification performance, enhancing the efficacy of intrusion detection across diverse network architectures.

This thesis also makes significant contributions to the field of global resilience against cyber threats, as emphasized by SGD 16. As cyber threats continue to evolve and proliferate, the need for strong and adaptable cybersecurity solutions becomes increasingly urgent on a global scale. By offering innovative methodologies and empirical insights into enhancing network security and intrusion detection, this thesis contributes to fortifying digital infrastructure and fostering global resilience against cyber threats.

In conclusion, the research outlined in this thesis not only advances the field of cybersecurity but also holds broader implications for societal well-being and global sustainability by mitigating the risks posed by cyber threats and safeguarding critical digital infrastructure.

4.3 Future Prospects

The proposed approach in this thesis, while effective, comes with certain limitations that need to be addressed in future research. Despite the promising outcomes achieved by the proposed ABC-ANN methods, some limitations must be addressed in future research. One of the primary challenges lies in the trade-off between computational complexity and model performance. High-performance models with greater accuracy demand substantial computational resources, particularly during the Bayesian optimization step used for detecting intrusions. Consequently, future work should explore additional and more advanced hardware resources to overcome these challenges.

Additionally, there are aspects of the DAE-based ABC-ANN that can be further improved. Investigating hybrid models could enhance anomaly detection performance. By incorporating the ABC algorithm for tuning hyperparameters in the proposed method, parameters and ANN weights can be optimized simultaneously. This dual optimization process would contribute to the enhancement of the overall methodology.

In summary, future prospects for this research include:

1. Overcoming hardware limitations by exploring additional and more advanced computational resources.
2. Integrating automated hyperparameter tuning techniques within the ABC-ANN method to optimize parameters and ANN weights more effectively.

By focusing on these areas, future research can expand upon the groundwork established in this thesis, advancing the creation of more robust and efficient intrusion detection systems to enhance the protection of digital infrastructures globally.

BIBLIOGRAPHY

[1] Z. Ahmad, A. S. Khan, C. W. Shiang, J. Abdullah, and F. Ahmad, "Network intrusion detection system: A systematic study of machine learning and deep learning approaches," *Wiley Online LibraryZ Ahmad, A Shahid Khan, C Wai Shiang, J Abdullah, F AhmadTransactions on Emerging Telecommunications Technologies, 2021•Wiley Online Library*, vol. 32, no. 1, Jan. 2021, doi: 10.1002/ett.4150.

[2] H. Hacilar, Z. Aydin, and V. Ç. Güngör, "Network intrusion detection based on machine learning strategies: performance comparisons on imbalanced wired, wireless, and software-defined networking (SDN) network traffics," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 32, no. 4, pp. 623–640, Jul. 2024, doi: 10.55730/1300-0632.4091.

[3] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," *Proceedings of the International Joint Conference on Neural Networks*, pp. 1322–1328, 2008, doi: 10.1109/IJCNN.2008.4633969.

[4] P. Branco, L. Torgo, and R. P. Ribeiro, "A survey of predictive modeling on imbalanced domains," *ACM Comput Surv*, vol. 49, no. 2, Aug. 2016, doi: 10.1145/2907070.

[5] 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, pp. 321–357, 2002, doi: 10.1613/JAIR.953.

[6] S. Potluri, C. D.-2016 I. 21st international conference, and undefined 2016, "Accelerated deep neural networks for enhanced intrusion detection system," *ieeexplore.ieee.orgS Potluri, C Diedrich2016 IEEE 21st international conference on emerging technologies, 2016•ieeexplore.ieee.org*, Accessed: May 16, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7733515/

[7] V. Canh Vu and T. H. Hoang, "Detect Wi-Fi Network Attacks Using Parallel Genetic Programming," *Proceedings of 2018 10th International Conference on Knowledge and Systems Engineering, KSE 2018*, pp. 370–375, Dec. 2018, doi: 10.1109/KSE.2018.8573378.

[8] B. Kolukisa, B. K. 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, Apr. 2024, doi: 10.1016/J.CSI.2023.103808.

[9] J. Kevric, S. Jukic, A. S.-N. C. and Applications, and undefined 2017, "An effective combining classifier approach using tree algorithms for network intrusion detection," *SpringerJ Kevric, S Jukic, A SubasiNeural Computing and Applications, 2017•Springer*, vol. 28, pp. 1051–1058, Dec. 2017, doi: 10.1007/s00521-016-2418-1.

[10] A. Pattawaro and C. Polprasert, "Anomaly-Based Network Intrusion Detection System through Feature Selection and Hybrid Machine Learning Technique," *International Conference on ICT and Knowledge Engineering*, vol. 2018-November, pp. 64–69, Jul. 2018, doi: 10.1109/ICTKE.2018.8612331.

[11] F. Vaca, Q. N.-2018 I. 17th international symposium on, and undefined 2018, "An ensemble learning based wi-fi network intrusion detection system (wnids)," *ieeexplore.ieee.orgFD Vaca, Q Niyaz2018 IEEE 17th international symposium*

on network Computing and, 2018•ieeexplore.ieee.org, Accessed: May 16, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8548315/

[12] S. Dhaliwal, A. Nahid, R. A.- Information, and undefined 2018, "Effective intrusion detection system using XGBoost," *mdpi.comSS Dhaliwal, AA Nahid, R AbbasInformation, 2018•mdpi.com*, Accessed: May 16, 2024. [Online]. Available: https://www.mdpi.com/2078-2489/9/7/149

[13] G. Logeswari, S. Bose, & T. A.-I. A., and undefined 2023, "An intrusion detection system for sdn using machine learning," *pdfs.semanticscholar.orgG Logeswari, S Bose, T AnithaIntelligent Automation & Soft Computing, 2023•pdfs.semanticscholar.org*, vol. 35, no. 1, pp. 867–880, 2023, doi: 10.32604/iasc.2023.026769.

[14] A. Javaid, Q. Niyaz, W. Sun, M. A.-P. of the 9th EAI, and undefined 2016, "A deep learning approach for network intrusion detection system," *dl.acm.orgA Javaid, Q Niyaz, W Sun, M AlamProceedings of the 9th EAI International Conference on Bio-inspired, 2016•dl.acm.org*, 2016, doi: 10.4108/eai.3-12-2015.2262516.

[15] B. Zong *et al.*, "Deep autoencoding gaussian mixture model for unsupervised anomaly detection," *openreview.netB Zong, Q Song, MR Min, W Cheng, C Lumezanu, D Cho, H ChenInternational conference on learning representations, 2018•openreview.net*, Accessed: May 16, 2024. [Online]. Available: https://openreview.net/forum?id=BJJLHbb0-

[16] R. Aygun, … A. Y. on cyber security and cloud, and undefined 2017, "Network anomaly detection with stochastically improved autoencoder based models," *ieeexplore.ieee.orgRC Aygun, AG Yavuz2017 IEEE 4th international conference on cyber security and cloud, 2017•ieeexplore.ieee.org*, Accessed: May 16, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7987197/

[17] M. Yousefi-Azar, V. Varadharajan, L. Hamey, and U. Tupakula, "Autoencoder-based feature learning for cyber security applications," *ieeexplore.ieee.orgM Yousefi-Azar, V Varadharajan, L Hamey, U Tupakula2017 International joint conference on neural networks (IJCNN), 2017•ieeexplore.ieee.org*, 2017, doi: 10.1109/IJCNN.2017.7966342.

[18] B. Zhang, Y. Yu, J. L.-2018 I. I. C. on, and undefined 2018, "Network intrusion detection based on stacked sparse autoencoder and binary tree ensemble method," *ieeexplore.ieee.orgB Zhang, Y Yu, J Li2018 IEEE International Conference on Communications Workshops, 2018•ieeexplore.ieee.org*, Accessed: May 16, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8403759/

[19] P. Dahiya, D. S.-P. computer science, and undefined 2018, "Network intrusion detection in big dataset using spark," *ElsevierP Dahiya, DK SrivastavaProcedia computer science, 2018•Elsevier*, Accessed: May 16, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050918309037

[20] C. Wheelus, E. Bou-Harb, X. Z.-2018 I. International, and undefined 2018, "Tackling class imbalance in cyber security datasets," *ieeexplore.ieee.orgC Wheelus, E Bou-Harb, X Zhu2018 IEEE International Conference on Information Reuse and, 2018•ieeexplore.ieee.org*, 2018, doi: 10.1109/IRI.2018.00041.

[21] R. Abdulhammed, M. Faezipour, … A. A.-I. sensors, and undefined 2018, "Deep and machine learning approaches for anomaly-based intrusion detection of imbalanced network traffic," *ieeexplore.ieee.orgR Abdulhammed, M Faezipour, A Abuzneid, A AbuMallouhIEEE sensors letters, 2018•ieeexplore.ieee.org*,

Accessed: May 16, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8526292/

[22] J. Ran, Y. Ji, B. T.-2019 I. 89th vehicular technology, and undefined 2019, "A semi-supervised learning approach to ieee 802.11 network anomaly detection," *ieeexplore.ieee.orgJ Ran, Y Ji, B Tang2019 IEEE 89th vehicular technology conference (VTC2019-Spring), 2019•ieeexplore.ieee.org*, Accessed: May 16, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8746576/

[23] A. Abdelkhalek and M. Mashaly, "Addressing the class imbalance problem in network intrusion detection systems using data resampling and deep learning," *Journal of Supercomputing*, vol. 79, no. 10, pp. 10611–10644, Jul. 2023, doi: 10.1007/S11227-023-05073-X.

[24] M. Imran, N. Haider, M. Shoaib, I. R.-C. and Electrical, and undefined 2022, "An intelligent and efficient network intrusion detection system using deep learning," *ElsevierM Imran, N Haider, M Shoaib, I RazzakComputers and Electrical Engineering, 2022•Elsevier*, Accessed: May 16, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045790622000684

[25] S. Gupta, M. Tripathi, J. G.-C. and E. Engineering, and undefined 2022, "Hybrid optimization and deep learning based intrusion detection system," *ElsevierSK Gupta, M Tripathi, J GroverComputers and Electrical Engineering, 2022•Elsevier*, Accessed: May 16, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045790622001653

[26] E. Qazi, M. Faheem, T. Z.-A. Sciences, and undefined 2023, "HDLNIDS: hybrid deep-learning-based network intrusion detection system," *mdpi.comEUH Qazi, MH Faheem, T ZiaApplied Sciences, 2023•mdpi.com*, Accessed: May 16, 2024. [Online]. Available: https://www.mdpi.com/2076-3417/13/8/4921

[27] J. Bhayo, S. Shah, S. Hameed, A. Ahmed, … J. N.-… A. of A., and undefined 2023, "Towards a machine learning-based framework for DDOS attack detection in software-defined IoT (SD-IoT) networks," *ElsevierJ Bhayo, SA Shah, S Hameed, A Ahmed, J Nasir, D DraheimEngineering Applications of Artificial Intelligence, 2023•Elsevier*, Accessed: May 16, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197623006164

[28] O. Tayfour, O. E. Tayfour, · Muhammad, and N. Marsono, "Collaborative detection and mitigation of DDoS in software-defined networks," *SpringerOE Tayfour, MN MarsonoThe Journal of Supercomputing, 2021•Springer*, vol. 77, no. 11, pp. 13166–13190, Nov. 1234, doi: 10.1007/s11227-021-03782-9.

[29] S. Yoo, S. Kim, S. Kim, B. K.-I. Sciences, and undefined 2021, "AI-HydRa: Advanced hybrid approach using random forest and deep learning for malware classification," *ElsevierS Yoo, S Kim, S Kim, BB KangInformation Sciences, 2021•Elsevier*, Accessed: May 16, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025520308525

[30] M. S. Elsayed, N. Le-Khac, … S. D.-P. of the 16th, and undefined 2020, "Network anomaly detection using LSTM based autoencoder," *dl.acm.orgM Said Elsayed, NA Le-Khac, S Dev, AD JurcutProceedings of the 16th ACM Symposium on QoS and Security for Wireless and, 2020•dl.acm.org*, pp. 37–45, Nov. 2020, doi: 10.1145/3416013.3426457.

[31] D. Firdaus, R. Munadi, Y. P.-2020 3rd International, and undefined 2020, "DDoS attack detection in software defined network using ensemble k-means++ and random forest," *ieeexplore.ieee.orgD Firdaus, R Munadi, Y Purwanto2020 3rd*

International Seminar on Research of Information, 2020•ieeexplore.ieee.org, doi: 10.1109/ISRITI51436.2020.9315521.

[32] M. S. Elsayed, H. Z. Jahromi, M. M. Nazir, and A. D. Jurcut, "The Role of CNN for Intrusion Detection Systems: An Improved CNN Learning Approach for SDNs," *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, vol. 382, pp. 91–104, 2021, doi: 10.1007/978-3-030-78459-1_7.

[33] Q. Tian *et al.*, "An intrusion detection approach based on improved deep belief network," *SpringerQ Tian, D Han, KC Li, X Liu, L Duan, A CastiglioneApplied Intelligence, 2020•Springer*, vol. 50, no. 10, pp. 3162–3178, Oct. 2020, doi: 10.1007/s10489-020-01694-4.

[34] D. Jing and H. B. Chen, "SVM based network intrusion detection for the UNSW-NB15 dataset," *Proceedings of International Conference on ASIC*, Oct. 2019, doi: 10.1109/ASICON47005.2019.8983598.

[35] R. Kumar, A. Malik, and V. Ranga, "An intellectual intrusion detection system using Hybrid Hunger Games Search and Remora Optimization Algorithm for IoT wireless networks," *Knowl Based Syst*, vol. 256, Nov. 2022, doi: 10.1016/J.KNOSYS.2022.109762.

[36] M. S. Elsayed, N. A. Le-Khac, and A. D. Jurcut, "InSDN: A novel SDN intrusion dataset," *IEEE Access*, vol. 8, pp. 165263–165284, 2020, doi: 10.1109/ACCESS.2020.3022633.

[37] U. Ali, K. K. Dewangan, and D. K. Dewangan, "Distributed denial of service attack detection using ant bee colony and artificial neural network in cloud computing," in *Nature inspired computing*, Springer, 2018, pp. 165–175.

[38] D. Karaboga and B. Akay, "Artificial bee colony (ABC) algorithm on training artificial neural networks," in *2007 IEEE 15th Signal Processing and Communications Applications*, 2007, pp. 1–4.

[39] C. Ozturk and D. Karaboga, "Hybrid artificial bee colony algorithm for neural network training," in *2011 IEEE congress of evolutionary computation (CEC)*, 2011, pp. 84–88.

[40] C. Ozkan, O. Kisi, and B. Akay, "Neural networks with artificial bee colony algorithm for modeling daily reference evapotranspiration," *Irrig Sci*, vol. 29, no. 6, pp. 431–441, 2011.

[41] G. Zhou, H. Moayedi, M. Bahiraei, and Z. Lyu, "Employing artificial bee colony and particle swarm techniques for optimizing a neural network in prediction of heating and cooling loads of residential buildings," *J Clean Prod*, vol. 254, p. 120082, 2020.

[42] S. Anuar, A. Selamat, and R. Sallehuddin, "Hybrid artificial neural network with artificial bee colony algorithm for crime classification," in *Computational Intelligence in Information Systems: Proceedings of the Fourth INNS Symposia Series on Computational Intelligence in Information Systems (INNS-CIIS 2014)*, 2015, pp. 31–40.

[43] H. Jahangir and D. R. Eidgahee, "A new and robust hybrid artificial bee colony algorithm–ANN model for FRP-concrete bond strength evaluation," *Compos Struct*, vol. 257, p. 113160, 2021.

[44] K. Taheri, M. Hasanipanah, S. B. Golzar, and M. Z. A. Majid, "A hybrid artificial bee colony algorithm-artificial neural network for forecasting the blast-produced ground vibration," *Eng Comput*, vol. 33, no. 3, pp. 689–700, 2017.

[45] P. G. Asteris and M. Nikoo, “Artificial bee colony-based neural network for the prediction of the fundamental period of infilled frame structures,” *Neural Comput Appl*, vol. 31, no. 9, pp. 4837–4847, 2019.
[46] B. Hajimirzaei and N. J. Navimipour, “Intrusion detection for cloud computing using neural networks and artificial bee colony optimization algorithm,” *Ict Express*, vol. 5, no. 1, pp. 56–59, 2019.
[47] M. S. Mahmod, Z. A. H. Alnaish, and I. A. A. Al-Hadi, “Hybrid intrusion detection system using artificial bee colony algorithm and multi-layer perceptron,” *International Journal of Computer Science and Information Security*, vol. 13, no. 2, p. 1, 2015.
[48] G. Pang, C. Shen, L. Cao, and A. Van Den Hengel, “Deep learning for anomaly detection: A review,” *ACM computing surveys (CSUR)*, vol. 54, no. 2, pp. 1–38, 2021.
[49] D. Javaheri, S. Gorgin, J.-A. Lee, and M. Masdari, “Fuzzy logic-based DDoS attacks and network traffic anomaly detection methods: Classification, overview, and future perspectives,” *Inf Sci (N Y)*, vol. 626, pp. 315–338, 2023.
[50] M. Jain, G. Kaur, and V. Saxena, “A K-Means clustering and SVM based hybrid concept drift detection technique for network anomaly detection,” *Expert Syst Appl*, vol. 193, p. 116510, 2022.
[51] Y. Zhong *et al.*, “HELAD: A novel network anomaly detection model based on heterogeneous ensemble learning,” *Computer Networks*, vol. 169, p. 107049, 2020.
[52] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, “Kitsune: an ensemble of autoencoders for online network intrusion detection,” *arXiv preprint arXiv:1802.09089*, 2018.
[53] A. Chen, Y. Fu, X. Zheng, and G. Lu, “An efficient network behavior anomaly detection using a hybrid DBN-LSTM network,” *Comput Secur*, vol. 114, p. 102600, 2022.
[54] A. Deng and B. Hooi, “Graph neural network-based anomaly detection in multivariate time series,” in *Proceedings of the AAAI conference on artificial intelligence*, 2021, pp. 4027–4035.
[55] K. Ding, Q. Zhou, H. Tong, and H. Liu, “Few-shot network anomaly detection via cross-network meta-learning,” in *Proceedings of the Web Conference 2021*, 2021, pp. 2448–2456.
[56] H. Najafi Mohsenabad and M. A. Tut, “Optimizing Cybersecurity Attack Detection in Computer Networks: A Comparative Analysis of Bio-Inspired Optimization Algorithms Using the CSE-CIC-IDS 2018 Dataset,” *Applied Sciences*, vol. 14, no. 3, p. 1044, 2024.
[57] P. Sanju, “Enhancing intrusion detection in IoT systems: A hybrid metaheuristics-deep learning approach with ensemble of recurrent neural networks,” *Journal of Engineering Research*, vol. 11, no. 4, pp. 356–361, 2023.
[58] A. A. E.-B. Donkol, A. G. Hafez, A. I. Hussein, and M. M. Mabrook, “Optimization of intrusion detection using likely point PSO and enhanced LSTM-RNN hybrid technique in communication networks,” *IEEE Access*, vol. 11, pp. 9469–9482, 2023.
[59] M. Kaveh and M. S. Mesgari, “Application of meta-heuristic algorithms for training neural networks and deep learning architectures: A comprehensive review,” *Neural Process Lett*, pp. 1–104, 2022.
[60] W. A. H. M. Ghanem, A. Jantan, S. A. A. Ghaleb, and A. B. Nasser, “An efficient intrusion detection model based on hybridization of artificial bee colony

and dragonfly algorithms for training multilayer perceptrons," *IEEE Access*, vol. 8, pp. 130452–130475, 2020.
[61] L. Karuppusamy, J. Ravi, M. Dabbu, and S. Lakshmanan, "Chronological salp swarm algorithm based deep belief network for intrusion detection in cloud using fuzzy entropy," *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields*, vol. 35, no. 1, p. e2948, 2022.
[62] J. Ahmad, S. A. Shah, S. Latif, F. Ahmed, Z. Zou, and N. Pitropakis, "DRaNN_PSO: A deep random neural network with particle swarm optimization for intrusion detection in the industrial internet of things," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 10, pp. 8112–8121, 2022.
[63] W. Elmasry, A. Akbulut, and A. H. Zaim, "Evolving deep learning architectures for network intrusion detection using a double PSO metaheuristic," *Computer Networks*, vol. 168, p. 107042, 2020.
[64] P. R. Kanna and P. Santhi, "Hybrid intrusion detection using mapreduce based black widow optimized convolutional long short-term memory neural networks," *Expert Syst Appl*, vol. 194, p. 116545, 2022.
[65] S. Saif, P. Das, S. Biswas, M. Khari, and V. Shanmuganathan, "HIIDS: Hybrid intelligent intrusion detection system empowered with machine learning and metaheuristic algorithms for application in IoT based healthcare," *Microprocess Microsyst*, p. 104622, 2022.
[66] R. Ghanbarzadeh, A. Hosseinalipour, and A. Ghaffari, "A novel network intrusion detection method based on metaheuristic optimisation algorithms," *J Ambient Intell Humaniz Comput*, vol. 14, no. 6, pp. 7575–7592, 2023.
[67] A. A. Malibari *et al.*, "A novel metaheuristics with deep learning enabled intrusion detection system for secured smart environment," *Sustainable Energy Technologies and Assessments*, vol. 52, p. 102312, 2022.
[68] A. Ponmalar and V. Dhanakoti, "Hybrid Whale Tabu algorithm optimized convolutional neural network architecture for intrusion detection in big data," *Concurr Comput*, vol. 34, no. 19, p. e7038, 2022.
[69] C. M. Bishop, *Neural networks for pattern recognition*, vol. 1. Oxford University Press., 1995.
[70] R. Nishino and S. H. C. Loomis, "Cupy: A numpy-compatible library for nvidia gpu calculations," *31st confernce on neural information processing systems*, vol. 151, no. 7, 2017.
[71] B. Komer, J. Bergstra, and C. Eliasmith, "Hyperopt-sklearn," *Automated Machine Learning: Methods, Systems, Challenges*, pp. 97–111, 2019.
[72] M. Sarhan, S. Layeghy, and M. Portmann, "Towards a standard feature set for network intrusion detection system datasets," *Mobile networks and applications*, pp. 1–14, 2022.
[73] O. Rainio, J. Teuho, and R. Klén, "Evaluation metrics and statistical tests for machine learning," *Sci Rep*, vol. 14, no. 1, p. 6086, 2024.
[74] A. F. Gad, "PyGAD: An Intuitive Genetic Algorithm Python Library," 2021.
[75] L. J. V Miranda, "PySwarms, a research-toolkit for Particle Swarm Optimization in Python," *J Open Source Softw*, vol. 3, no. 21, 2018, doi: 10.21105/joss.00433.
[76] Martín~Abadi *et al.*, " TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," 2015. [Online]. Available: https://www.tensorflow.org/
[77] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[78] G. Van Rossum, F. L. Drake, and others, “Python 3 Reference Manual:(Python Documentation Manual Part 2),” *Scotts Valley, CA: CreateSpace*, 2009.

APPENDIX A

Table A. 1 Selected features and their descriptions of the AWID dataset

Selected Features	Descriptions
frame.time_epoch	Epoch Time
radiotap.datarate	Data rate (Mb/s)
frame.len	Frame length on the wire
frame.time_delta	Time delta from previous captured frame
wlan.fc.type	Types of 802.11 packets
radiotap.dbm_antsignal	Antenna signal
wlan.fc.subtype	Specific types of 802.11 packets
wlan.fc.frag	Control bit for more fragments coming or not
wlan.fc.retry	Control bit for frame is a retransmission or not
wlan.fc.pwrmgt	Control bit for station will stay awake or sleep
wlan.fc.protected	Control bit for MSDU payload encrypted or not
wlan.duration	Duration
frame.time_delta_displayed	Time delta from previous displayed frame
frame.time_relative	Time since reference or first frame
radiotap.channel.type.cck	Complementary Code Keying (CCK)
radiotap.channel.type.ofdm	Orthogonal Frequency-Division Multiplexing (OFDM)
frame.cap_len	Frame length stored into the capture file
radiotap.mactime	MAC timestamp
radiotap.channel.freq	Channel frequency
wlan.fc.moredata	Frame Control Field More Data

Table A. 2 Selected features and their descriptions of the UNSW-NB15 dataset

Selected Features	Descriptions
dur	Record total duration
proto=tcp	Transaction protocol
proto=arp	Transaction protocol
proto=ospf	Transaction protocol
proto=unas	Transaction protocol
service=-	http, ftp, ssh, dns ..,else (-)
service=http	http, ftp, ssh, dns ..,else (-)
service=dns	http, ftp, ssh, dns ..,else (-)
service=ftp	http, ftp, ssh, dns ..,else (-)
state=CON	The state and its dependent protocol, e.g. ACC, CLO, else

	(-)
spkts	Source to destination packet count
dbytes	Destination to source bytes
sbytes	Source to destination bytes
rate	there is no description
sload	Source bits per second
dttl	Destination to source time to live
Sttl	Source to destination time to live
synack	The time between the SYN and the SYN_ACK packets of the TCP
dload	Destination bits per second
dloss	Destination packets retransmitted or dropped
sloss	Source packets retransmitted or dropped
sinpkt	Source inter-packet arrival time (mSec)
dtcpb	Destination TCP base sequence number
stcpb	Source TCP base sequence number
tcprtt	The sum of 'synack' and 'ackdat' of the TCP
smean	Mean of the flow packet size transmitted by the src
dmean	Mean of the flow packet size transmitted by the dst
trans_depth	the depth into the connection of http request/response transaction
response_body_len	The content size of the data transferred from the server's http service.
ct_srv_src	Number of connections that contain the same service and source address in 100 connections according to the last time.
ct_dst_ltm	Number of connections of the same destination address in 100 connections according to the last time.
ct_src_dport_ltm	Number of connections of the same source address and the destination port in 100 connections according to the last time.
ct_dst_sport_ltm	Number of connections of the same destination address and the source port in 100 connections according to the last time.
ct_state_ttl	Number for each state according to specific range of values for source/destination time to live.
ct_dst_src_ltm	Number of connections of the same source and the destination address in 100 connections according to the last time.
ct_srv_dst	Number of connections that contain the same service and destination address in 100 connections according to the last time .
ct_src_ltm	Number of connections of the same source address in 100 connections according to the last time.

CURRICULUM VITAE

2008 – 2012	B.Sc., Computer Engineering, Erciyes University, Kayseri, TURKEY
2017 – 2024	Ph.D., Electrical and Computer Engineering, Abdullah Gül University, Kayseri, TURKEY
2020 – Present	Research Assistant, Computer Engineering, Abdullah Gül University, Kayseri, TURKEY

SELECTED PUBLICATIONS AND PRESENTATIONS

J1) Kolukısa, B., Hacılar, H., Kuş, M., Bakır-Güngör, B., Aral, A., & Güngör, V. Ç. (2019). Diagnosis of coronary heart disease via classification algorithms and a new feature selection methodology. *International Journal of Data Mining Science*, *1*(1), 8-15.

J2) Bakir-Gungor, B., Hacılar, H., Jabeer, A., Nalbantoglu, O. U., Aran, O., & Yousef, M. (2022). Inflammatory bowel disease biomarkers of human gut microbiota selected via different feature selection methods. *PeerJ*, *10*, e13205.

J3) Hacilar, H., Gormez, Y., Bakir-Gungor, B., Gezer, C., Aydin, Z., & Cagri, V. (2020). Classification of Cardiovascular Artery Diseases Using Artificial Neural Network. *International Journal of Bioscience, Biochemistry, and Bioinformatics*, *10*(1), 34-41.

J4) Kolukisa, B., Dedeturk, B. K., Hacilar, H., & Gungor, V. C. (2024). An efficient network intrusion detection approach based on logistic regression model and parallel artificial bee colony algorithm. *Computer Standards & Interfaces*, *89*, 103808.

J5) HACILAR, H., Aydin, Z. A. F. E. R., & GÜNGÖR, V. Ç. (2024). Network intrusion detection based on machine learning strategies: performance comparisons on imbalanced wired, wireless, and software-defined networking (SDN) network traffics. Turkish Journal of Electrical Engineering and Computer Sciences, 32(4), 623-640.

J6) Hacilar H., Dedeturk B.K., Gungor V.Ç. Network Anomaly Detection using Deep Autoencoder and Parallel Artificial Bee Colony Algorithm-Trained Neural Network. *PeerJ*, Accepted.

J7) Hacilar H., Dedeturk B.K., Ozmen M., Eraslan-Celik M.i Gungor V.Ç. Optimizing ANN with Parallel ABC Algorithm for Network Intrusion Detection and Earthquake Damage Assessment. *Expert Systems*, In Review.

C1) Kolukisa, B., Hacilar, H., Goy, G., Kus, M., Bakir-Gungor, B., Aral, A., & Gungor, V. C. (2018, December). Evaluation of classification algorithms, linear discriminant analysis and a new hybrid feature selection methodology for the diagnosis of coronary artery disease. In *2018 ieee international conference on big data (big data)* (pp. 2232-2238). IEEE.

C2) Hacılar, H., Nalbantoğlu, O. U., & Bakir-Güngör, B. (2018, September). Machine learning analysis of inflammatory bowel disease-associated metagenomics dataset. In *2018 3rd international conference on computer science and engineering (UBMK)* (pp. 434-438). IEEE.

C3) Coşkun, M., Hacılar, H., Gezer, C., & Gungor, V. C. (2019, June). OFFER Referees Suggester for the Journal Editors. In *2019 IEEE Symposium on Computers and Communications (ISCC)* (pp. 1-6). IEEE.

C4) Gülşen, A., Kolukısa, B., & Hacılar, H. (2024, September). A Federated Learning Framework for Ultrasonic Nondestructive Testing Image Classification, UBMK, Accepted.