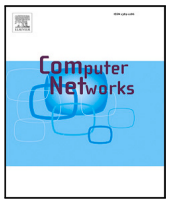
Computer Networks 271 (2025) 111600

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



On resource consumption of machine learning in communications network security

Md Muzammal Hoque [a,*], Ijaz Ahmad [a], Jani Suomalainen [a], Paolo Dini [b], Mohammad Tahir [c]

[a] VTT Technical Research Centre of Finland, FI-02044, Espoo, Finland
[b] CTTC/CERCA, 08860, Barcelona, Spain
[c] University of Turku, Department of Computing, FI-20014, Turku, Finland

ARTICLE INFO

Keywords:
Security
6G
Distributed security
Network security
Resource consumption
Resource efficiency
Machine learning
ML
6G security
Sustainability
DNN

ABSTRACT

As the complexity of communication networks continues to increase, driven by a diverse array of devices, services and applications, the adoption of Machine Learning (ML) has seen a significant rise to address various challenges ranging from management to security. Regarding network security, the application of ML ranges from preventive measures to detection and remediation due to its ability to dynamically learn and adapt to evolving threat landscapes. However, ML requires a significant amount of resources, mainly due to the fact that ML operates on data, and the volumes of data are consistently rising. This review article explores the resource consumption aspect of ML techniques used for network security and provides a comprehensive review of the current state of research. Moreover, we propose a taxonomy that can be used to classify the methods through which the resource consumption can be reduced for different ML-based network security implementations. The focus of the study encompasses several key aspects related to resource consumption, including energy, computing, memory, latency, bandwidth, and human resources. These resources are critical in improving the efficiency and optimizing the reliability and sustainability of network security solutions. Furthermore, based on an extensive literature review, we summarize key points regarding optimizing resource consumption in ML-based network security solutions. Finally, the challenges and future research directions for resource-efficient, ML-based network security solutions are outlined to aid in the advancement of research in this area.

1. Introduction

Communication networks, including wireless, fixed, and non-terrestrial networks, are integrating new and diverse technologies to meet the growing demands of emerging services and applications. The resulting integrated network architectures and infrastructures are highly complex. Moreover, network traffic continues to increase due to expanding user bases and novel services. For example, the expected data rate in sixth-generation (6G) wireless networks is anticipated to be one terabit per second, compared to 20 gigabits per second in fifth-generation (5G) wireless networks [1]. Therefore, managing the overall network architecture and infrastructure, while meeting changing user and service demands, has necessitated the adoption of Machine Learning (ML)-based network operations in an automated fashion [2]. Network security, due to the increasing complexity of overall network and traffic patterns, has become one of the foremost application areas of ML in communication networks.

The applications of ML have rapidly increased in communications networks [3,4], and the security of communications networks has become one of the most important application areas of ML [2], paving the way towards security automation [5]. Communications network security, generally, encompasses various preventive and responsive mechanisms that ensure the reliable and timely transmission of information between two devices connected through a communication medium. The main cause of the shift towards automated ML-based security is the radical growth in network traffic, end-user services and devices, and a huge range of critical infrastructures, rendering other approaches inadequate. Due to this, cybersecurity can be considered as the foremost important and, probably, the first application of ML or AI in communication networks. Starting with a simplistic use case of CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) [6] in communications security, from the late 1990s to early 2000s, ML has progressed towards resource-intensive

* Corresponding author.
E-mail addresses: muzammal.hoque@vtt.fi (M.M. Hoque), ijaz.ahmad@vtt.fi (I. Ahmad), jani.suomalainen@vtt.fi (J. Suomalainen), paolo.dini@cttc.es (P. Dini), tahir.mohammad@utu.fi (M. Tahir).

https://doi.org/10.1016/j.comnet.2025.111600
Received 19 March 2025; Received in revised form 30 June 2025; Accepted 31 July 2025
Available online 8 August 2025

Table 1
Expansion of acronyms and abbreviations.

Acronym	Expansion
3GPP	Third Generation Partnership Project
5G	Fifth Generation
6G	Sixth Generation
ABC	Ada Boost Classifier
ADAS	Advanced Driver Assistance Systems
AE	Autoencoder
AI	Artificial Intelligence
ANN	Artificial Neural Network
ARL	Adversarial Reinforcement Learning
AutoML	Automated ML
Bagging	Bagging Ensemble Learning
Boosting	Boosting Ensemble Learning
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DDoS	Distributed Denial of Service
DNN	Deep Neural Network
DQL	Deep Q-Learning
DRL	Deep Reinforcement Learning
DT	Decision Tree
DVFS	Dynamic Voltage and Frequency Scaling
EI	Edge Intelligence
ENR	Elastic Net Regression
ETSI	European Telecommunications Standards Institute
ETC	Extra Trees Classifier
FL	Federated Learning
GBC	Gradient Boosting Classifier
GPU	Graphics Processing Unit
GPR	Gaussian Process Regression
GRU	Gated Recurrent Units
HPO	Hyperparameter Optimization
ICT	Information and Communications Technology
IDS	Intrusion Detection System
IoT	Internet of Things
IT	Information Technology
KPI	Key Performance Indicator
KR	Kernel Regression
KNN	K-Nearest Neighbours
LDA	Linear Discriminant Analysis
Lasso	Lasso Regression
LGBM	Light Gradient Boosting Machine
LSTM	Long Short-Term Memory
MI	Mutual Information
ML	Machine Learning
MLP	Multi-Layer Perceptron
NB	Naïve Bayes
NIDS	Network Intrusion Detection System
NN	Neural Network
PART	Partial Decision Tree
PCA	Principal Component Analysis
QDA	Quadratic Discriminant Analysis
RAN	Radio Access Network
RC	Ridge Classifier
RF	Random Forest
RNN	Recurrent Neural Network
RR	Ridge Regression
SVM	Support Vector Machine
SVR	Support Vector Regression
VAE	Variational Autoencoder
XAI	Explainable AI
XGB	eXtreme Gradient Boosting
ZTNs	Zero-Touch Networks

security applications, such as intrusion detection with deep learning techniques for deep packet inspection [7], and intent and log analysis with Large Language Models (LLMs) [8] in communications networks.

To meet the needs of different use cases and applications, ML algorithms are progressing towards larger and computationally complex models, which require higher computational and energy resources [9]. Resource consumption of ML techniques, generally, has not been a major concern while developing ML algorithms, since the major focus has been on model accuracy and efficiency. However, the consumption of various resources by ML models has become a major concern recently, since it has a significant impact on both economic and environmental costs [10]. For example, the growth rate of energy consumption by AI or ML systems is estimated to be around 30% annually [11], and the computational costs of ML, specifically deep learning, have increased nearly 300,000x in six years, leading to very high carbon footprints [12]. Such an increase in the consumption of resources also leads to high capital and operational costs for communications networks.

In communication networks, the critical challenge of ensuring the availability of underlying resources to execute ML techniques, from training to inference, is largely overlooked in state-of-the-art research. This article, therefore, focuses on the application of ML in the security of communication networks, with particular emphasis on the resource consumption of ML techniques and procedures used in network security. We explore the main resources necessary for ML operations in network security, such as energy, processing power, memory, time, bandwidth, and human involvement. We then conduct a comprehensive examination of various ML applications within the realm of communication network security, including intrusion detection systems, anomaly detection, and threat prediction and prevention. We investigate how these applications leverage ML to enhance security measures while studying resource occupancy. Additionally, the article discusses the trade-offs between security and resource efficiency, providing insights into optimizing ML models for deployment in future communication networks. We also highlight important research gaps and offer interesting insights to inspire future research on resource-efficient and resource-aware ML-based network security.

Fingerprinting the resource consumption of ML is generally not straightforward, since there are many trade-offs, such as efficiency in terms of resources versus accuracy or the required precision. Network security further complicates this trade-off, due to the many variables or conflicting key performance indicators (KPIs) for resource efficiency versus desired level of protection. Therefore, we carefully selected the KPIs and organized the procedures used for resource optimization in a detailed taxonomy to provide a coherent and subjective analysis. In this vein, the main contributions of the article are as follows:

- Identify the main resources and their trade-offs for ML techniques used for the security of communication networks.
- Investigate ML techniques that consume lower resources and their implications for achieving resource efficiency in terms of the target security parameters, such as accuracy and precision, and detecting false positives and false negatives.
- Discussion on the security implications and trade-offs, with respect to resource efficiency, to derive important future research directions.
- Key research directions on improving network security using ML without over-using or exhausting network resources.

Most existing surveys and reviews focus on improving the resource efficiency of either specific ML techniques or their specific use cases or applications, as presented in the related work. However, until the time of this writing, no work has focused on exploring the resource consumption of ML techniques used for the security of communication networks. This article is organized as follows: In Section 2, we discuss the background of using ML in communications networks and for the security of communications networks. Section 3 describes the relevant KPIs and taxonomy on which the rest of the discussion in the article is based. Section 4 discusses different solutions for network security using ML that reduce different resource usage, based on the KPIs and taxonomy outlined in Section 3. Section 5 presents key lessons learned based on the extensive analysis of resource consumption using ML techniques for network security. Section 6 highlights the main existing challenges and outlines important future research directions. This article is concluded in Section 7. For clarity, the organization of the article is also presented in Fig. 1 (see Table 1).

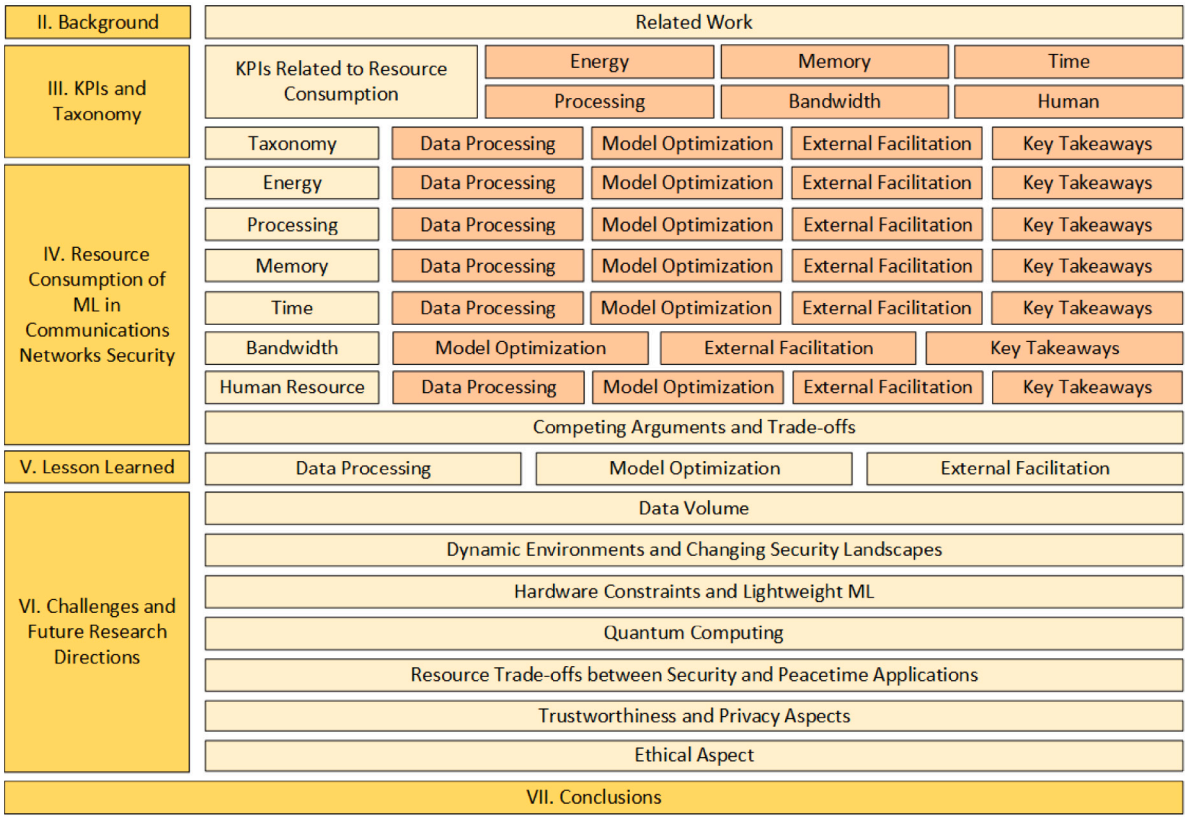


Fig. 1. Overview of the article organization.

2. Background

The security threat landscape of communications networks has been evolving, more so in the wireless networks from the first generation (1G) of wireless networks to 5G, with an increasing threat landscape, from simple tuning into frequency channels to complex attacks on critical infrastructures [13]. To make sense of the huge amounts of traffic in transit in communications networks and complex environments, ML has been widely researched for security threat detection, prevention, and mitigation [7,14,15]. Due to its vast application areas, also beyond security, embedding ML in the communications networks architecture has been proposed and standardized, for example, by the European Telecommunications Standards Institute (ETSI) and the Third Generation Partnership Project (3GPP) [16]. The 6G native-AI has been a hot topic in terms of research and funding, such as from the European Union Horizon Europe initiatives, where ML-based security has been considered integral for end-to-end network security. To understand the scope and dimensions of security, we discuss communications network security to be comprised of the following generalized security dimensions by the International Telecommunication Union-Telecommunications (ITU-T) recommendations [17,18]:

- **Access control:** Access control ensures that only authorized devices or personnel can access network elements, services, and stored information by preventing unauthorized access.
- **Authentication:** Authentication confirms the identity of communication entities and ensures their validity.
- **Non-repudiation:** Non-repudiation prevents the denial of responsibility for any particular action performed by any entity through providing the proper proof of distinct network-centric actions.
- **Data confidentiality:** Data confidentiality ensures the non-disclosure of data to prevent unauthorized entities from reading or modifying the data in transit.
- **Communication security:** This dimension ensures the security of the information flow so that it travels through only authorized endpoints and is not intercepted in between.
- **Data integrity:** Data integrity ensures unauthorized personnel have not modified, replicated, created, or deleted the data.
- **Availability:** This dimension ensures the availability of stored information, services, network elements, and applications to authorized users, even after any event impacting the network occurs.
- **Privacy:** Privacy ensures that the user information and activities are not extracted from network traffic.

These dimensions are important for the overall communications security landscape. Different security techniques are deployed in the realms of detection, protection, prevention, and remediation to cover these dimensions. Recently, common to these realms of security are data-driven approaches using ML to automatically understand the complexities of communication and networking environments and dynamically adjust protocols without the need for human intervention [19]. Moreover, there are specific security applications and use cases of communications or network security, such as intrusion detection and prevention systems (IDS/IPS), ingress port scanning, DoS and DDoS detection, and anomaly detection, to name a few, where ML has become inevitable [15]. Therefore, there is significant research ongoing on the implementation and improvement of ML-based techniques in the domain of security of communications networks. However, research literature investigating the additional costs of ML-based security techniques in communications networks remains scarce. In the following subsection, we discuss the related survey articles that discuss the resource consumption of ML or AI in the domain of network security.

2.1. Related work

The research efforts on ML, used in the fields of cybersecurity, security of communication networks, networking technologies, and

Table 2
Existing and recent survey articles on resource efficiency of ML.

Ref	Year	Main focus	E	P	M	T	B	H
[20]	2022	Resource-efficient distributed AI for IoT applications	✓	✓	✓	✓		
[21]	2019	Efficient deep learning in terms of quality and resources.			✓	✓		
[22]	2024	Methods and techniques for enhancing energy efficiency in FL systems	✓					
[23]	2022	Optimizing resource allocation in 6G networks to balance security and QoS.	✓	✓	✓	✓	✓	
[24]	2023	Optimizing the resource efficiency of CNNs	✓	✓	✓	✓		
[25]	2020	In-memory computing architectures for low latency and high computational demands of ML	✓	✓	✓			
[26]	2020	Communication-efficient techniques to address resource limitation for edge AI	✓	✓	✓	✓	✓	
[27]	2024	Red AI vs. Green AI: environmental impacts and potential solutions	✓					
[28]	2023	Edge AI for energy efficient autonomous driving services	✓					
[29]	2020	ML performance, energy, and speed trade-off in ADAS applications	✓		✓	✓		
[30]	2024	Lightweight DL for resource-constrained environments	✓	✓	✓	✓		✓
[31]	2023	ML and MEC integration for intelligent resource allocation and overcoming UAV resource constraints.	✓				✓	
[32]	2024	DL workload schedulers for GPU data centres to reduce costs and optimize resource use.	✓					
[33]	2020	ML for energy-efficient 5G network	✓					
[34]	2022	Energy utilization management strategies in cloud data centres	✓					
[35]	2023	Achievement of energy-aware security in IoT networks	✓					
[36]	2022	Importance of human engagement in the ML pipeline.						✓
[37]	2023	Examining strategies for integrating human input into the learning process.						✓
[38]	2019	Investigation and analysis of ML techniques used for IDS		✓	✓	✓		
[39]	2024	Physical Layer Authentication and Security Design with ML				✓		
Our work		Resource demands and minimization strategies of ML in network security	✓	✓	✓	✓	✓	✓

Bandwidth: B, Energy: E, Memory: M, Processing: P, Time: T, Human: H.

application and use cases, can be seen from the survey articles published on these topics in Table 2. Several survey articles discuss the resource efficiency of different ML techniques. However, most of these articles are not related to network security and mainly focus on a specific resource, such as energy or memory, or focus on a specific type of ML technique. For example, Danish et al. [22] conducted a survey that discusses techniques and methods used for energy efficiency enhancement of FL systems. The article elaborates on different algorithms, optimization, and resource allocation strategies. The authors also discussed and examined the role of emerging technologies in minimizing the energy footprint for FL systems, such as 6G networks and blockchain. Gaurav [21] conducted a comprehensive survey based on the common challenge of efficiency: keeping the training and inference fixed. The author provided a mental model for deep learning, consisting of efficiency matrices, five focus areas, and a comparison between models. The authors considered two types of matrices: quality matrices and footprint matrices. The quality matrices considered are the standard matrices, such as accuracy, precision, etc. In contrast, the footprint matrices consider RAM consumption, latency, model disk size, number of parameters, number of epochs to convergence, etc. The five key areas identified for developing smaller, faster, and better deep learning models are compression techniques, learning techniques, automation, efficient architectures, and infrastructure. The discussion included tools, algorithms, and infrastructure for efficient deep learning. The author also provided a practitioner guide to efficiency that provides valuable insight regarding the trade-off between footprint and model quality.

Emna et al. [20] surveyed the resource-efficient distributed AI, focused on pervasive computing. The discussion included the training, decision process, real-time training, and inference. It was initiated from the resource challenge for IoT application & communication and the importance of developing a pervasive AI system for this purpose. Fadlullah. et al. [23] attempted to address the research gap in balancing security level and quality of Services (QoS) in communication networks. The emerging 6G network will have to satisfy the demand for high-end service quality of the customers at the mobile edge users in terms of data rate and delay. Additionally, there will be a wide range of security threats like zero-day attacks and others. In most service provisioning techniques, security matrices are considered independently from Quality of Experience (QoE) and Quality of Service (QoS) parameters. Enhancing the security level using resource allocation optimization techniques often negatively impacts service quality. Here, experienced delay, data throughput, tolerable latency, resource utilization rate, jitters, energy, and spectral efficiency were considered as QoS/QoE parameters. Meanwhile, authentication strength, encryption key strength, privacy matrices, network anomaly detection rate, and others were considered as security parameters. The authors surveyed the existing practices, challenges, and opportunities regarding the 6G network, where various techniques are used, and found some AI-based techniques that balance security and service quality. The survey also included various types of techniques for QoS and/or Security, including convex optimization, matching theory, Graph-theory algorithms, stochastic optimization, queuing theory, AI-based learning techniques, Game-theoretic optimization, and others, but most of them do not satisfy both QoS and security concerns. There was also a discussion on several joint QoS-security techniques in various cases, like IoT, B5G, 5G, and 4G+, but the optimization techniques were not the same for each scenario. The authors then presented a Deep-learning-based QoS and security-level selection algorithm that can dynamically assign a security level to QoS parameters and provide a tunable security level.

JunKyu et al. [24] provided a survey covering the resource efficacy of Convolutional Neural Networks (CNNs) from model-level optimization, arithmetic optimization, and implementation-level optimization. The survey also identifies the research gaps for resource-efficient CNN in these three-level techniques. It also defines resource efficiency matrices for different techniques and clarifies the influence of the higher to lower-level techniques. Here, the model-level optimizer consists of the techniques of pruning, weight-average quantization, compact convolution, knowledge distillation, and neural architecture search. The arithmetic level consists of lower-precision inferences, encoding, using LUT, various formats, mixed-precision training, and BFP training. Finally, the implementation level consists of leveraging data reuse, sparsity-skipping, sparsity-encoding, sparsity decomposing, weight competition, innovative tech, as well as adapting early exiting, model switching, and channel width. Sathwika et al. [25] surveyed Non-Von Neumann computing architectures, which were introduced to overcome the lack of traditional computing hardware in processing resource-intensive algorithms, like ANN, DNN, and CNN. The authors focused on In-memory Computing (IMC)/ Processing-in-memory (PIM) and reviewed, analysed, and evaluated various state-of-the-art in-memory computing architectures from different dimensions and architectural aspects.

Yuanming et al. [26] conducted a comprehensive survey that describes different methods to overcome various communication challenges that arise in pushing training and inference of AI/ML in the edge nodes. The authors first identified the communication challenges in edge AI systems and then introduced various communication-efficient

techniques from both system and algorithmic perspectives for training and inference tasks at the edge. The considered challenges are resource limitation on the edge nodes, privacy and security constraints, and resource heterogeneity in the edge nodes. The authors summarized some communication-friendly algorithms for distributed AI model training on the edge nodes, including zeroth-order, first-order, second-order, and federated optimization algorithms. The authors then categorized various edge AI system architectures in data partition and model partition-based edge training systems. The authors also introduced general AI systems defined by edge computing and thoroughly discussed the communication challenges and solutions within these architectures.

Barbierato et al. [27] pointed out the downside of Red-AI, which prioritizes performance but consumes significant energy and recommended the potential solutions with "Green AI", which aims for efficiency and sustainability. The authors critically examined Red-AI, focusing on reducing computational complexity in model training through architectures, algorithms, and data structures. The authors also discussed the hardware and energy demands of intensive models and recent advancements in training optimizations. Finally, the authors highlighted Green AI as the emerging and essential approach to balancing performance and energy efficiency. Borrego-Carazo et al. [33] conducted a systematic review on how ML can be implemented in resource-constrained or embedded platforms devoted for Advanced Driver Assistance Systems (ADAS). The authors structured the collected information in five major ADAS tasks, including vehicle and pedestrian detection, driver's state, behaviour, and identification, traffic sign recognition, road segmentation and understanding, among others. Finally, the authors analysed the achievements of ML performance in terms of speed and energy trade-off. Katare et al. [28] conducted a survey on vehicular communication, applications, approximation, and edge AI techniques, focusing on energy efficiency of various approximation and enabling frameworks. The authors selected various AI and edge computing methods for driving services based on model size and real-time deployment in low-powered embedded devices. The aim was to find out the current trends and approaches that can support level 5 self-driving by enabling AI on edge devices, in an energy-efficient manner, as the answer to the primary question. The survey explores secondary questions related to model development, optimization, and inference techniques, including Federated Learning (FL).

Hou-I et al. [30] conducted a survey on various techniques that overcome the hardware constraints of DL without compromising the performance. The authors grouped lightweight architecture into a family series, such as MobileNet and ShuffleNet series, for improved clarity. The authors cover the light-weight DL from compression methods, such as pruning, quantization, knowledge distillation (KD), and neural architecture search (NAS). Additionally, they discussed hardware acceleration aspects for lightweight DL, such as hardware architectures, dataflow, and data locality optimization, deep learning libraries, and the co-design of hardware architecture. Ning et al. [31] conducted a comprehensive survey on the current techniques for ML and MEC in the internet of UAV and remarked that ML and MEC could complement each other in UAV applications. ML aids in intelligent resource allocation, task scheduling, and offloading decisions. Whereas, MEC addresses the limitation of UAV's resource constraints challenge in training and inference of ML. Two network architectures can be formed in this integration process, MEC-enabled ML and ML-enabled MEC. MEC-enabled ML allows ML to run on devices that do not have enough computational power; whereas, ML-enabled MEC helps in making smart decisions for offloading dynamically generated tasks. The authors presented some major challenges for UAVs on the internet and summarized the current solutions for those challenges. Those solutions were categorized as intelligent computation offloading, efficiency improvement, channel condition amelioration, load balance, multi-objective optimization, security and privacy, and neural layer offloading.

Zhisheng et al. [32] surveyed existing DL workload schedulers for GPU data centres. A GPU data centre scheduler has significant importance in reducing computational costs and proper resource utilization. The authors discussed the ways current schedulers manage different workloads, focusing on their scheduling goals and how they utilize resources, including scheduling DL training workload, inference workload, and other types of workloads. The authors remarked that the development of new schedulers is driven by DL achievements and user requirements. Additionally, sophisticated algorithms can significantly improve scheduling efficiency. Finally, new hardware resources can be used to create more efficient schedulers. Amna et al. [29] surveyed the state-of-the-art ML techniques in 5G, focusing on how these methods enhance energy efficiency across core, access, and edge networks. The authors also explored the enabling technologies within three networks, highlighting the role of machine learning in enhancing energy efficiency. Suraj et al. [34] compared various techniques for reducing energy consumption in cloud data centres, including ML, statistical methods, heuristics, and metaheuristics. They found that ML can reduce energy consumption by 1.6% to 88.5%. In contrast, statistical methods can achieve reductions of 5.4% to 84%, heuristics 5.4% to 90%, and metaheuristics 7.68% to 97%, depending on different benchmark approaches and settings. Michaël, Ghada, and Abdelmadjid [35] conducted a survey that addresses both security challenges and their effects on the energy efficiency of IoT networks. The authors classified the recent security solutions that reduce energy consumption in lightweight protocols, energy-efficient mechanisms, adaptive security, context-aware security, and energy-harvesting concepts for security. The authors highlighted this context-aware security as a viable strategy to enhance IoT network security, while reducing energy consumption. Xingjiao. et al. [36] conducted a comprehensive survey on human-in-the-loop (HITL) for ML, which is a crucial concept aimed at training an accurate prediction model with minimal cost by integrating human knowledge and experience. HITL discussed how human domain knowledge can promote the automation of ML from a data perspective. The survey focused on HITL works from a data processing perspective, model training perspective, and application perspective. Eduardo et al. [37] conducted a review on the state-of-the-art in Human-in-the-Loop (HITL) techniques, focusing on various strategies for incorporating humans into the learning process. These strategies include active learning (AL), where the system controls the learning process while relying on humans to provide annotations for unlabelled data; interactive machine learning (IML), where there is closer user-system interaction, with users providing information more frequently and incrementally than in traditional ML; and machine teaching (MT), where a human expert controls the learning process by defining the knowledge they want to transfer to the ML model. The authors defined various terms, clarified confusing or contradictory definitions, and determined the boundaries between different terms.

Mishra. et al. [38] conducted a comprehensive investigation to identify challenges associated with applying various machine learning techniques to intrusion detection systems (IDS). The authors reviewed different ML classifiers, evaluating their suitability, strengths, and limitations across various attack scenarios. Their analysis covered both single and multi-classifier approaches, including how different feature subsets influence classifier performance. They also highlighted the difficulty of detecting low-frequency attacks in network datasets using ML. While the study noted the memory, CPU, and time demands of certain ML techniques for IDS in various scenarios, it did not explore resource consumption in depth. Hoang et al. [39] conducted a comprehensive review of ML algorithms for physical layer security, highlighting ML's dual role in both authentication and secure transmission design. The authors also explored the use of NNs for solving stochastic optimization problems in this context. However, the study did not address resource optimization or the efficiency of ML algorithms.

There are various networking technologies that can help reducing the resource consumption of ML-based security techniques. For example, the edge–cloud continuum [40–42] can help resource-oriented

deployment of ML techniques, heavier models in centralized clouds, and lighter models being deployed in the edge. Similarly, SDN [18] improves resource optimization through programmability and global network resource visibility. Therefore, SDN-enabled ML deployment [43, 44] could help allocate network resource in an efficient manner. Similarly, network function virtualization (NFV) can help deploy network functions on run-time in different network perimeters based on resource-availability [45]. Furthermore, there are studies on platform-specific approaches where ML techniques are deployed in a resource-aware or resource-efficient manner, such as in the IoT domain [46]. However, there are no studies that focus on fingerprinting the needed resources for different ML-based security approaches in communications networks.

To the best of our knowledge, based on the existing literature, the work presented in this article is the first survey to focus on the resource consumption and efficiency of a wide range of ML techniques for network security solutions. This study focuses on different resources, including energy, processing, memory, time/latency, bandwidth, and human involvement, in the ML pipeline. Additionally, the discussion on reduction of resource consumption of ML techniques include ML algorithms, different ways through which resource consumption is reduced, the impact of varying sizes of network security datasets, and different tools that are used to measure the resource consumption. Finally, this work presents important future research directions regarding ML techniques and achieving resource efficiency in the domain of network security.

3. KPIs and taxonomy

The significant growth in the number of resource-constrained devices, like IoT, a wide array of mobile services, and the conflicting policies of the distinct mobile service operators and architectural evolution, make the security of the future network challenging [13]. Additionally, all the security domains defined by ITU-T have their individual implementation methods and resource costs. It would require specific modifications to reduce resource costs while maintaining the same level of security to achieve green security with them. ML-based security systems have the potential to meet the requirements of future 6G networks security challenges in many aspects, but their resource consumption will remain a key concern. Hence, resource-efficient methods need to be developed and require significant research efforts from the scientific community for the creation of reliable security systems for 6G networks [47]. This approach will not only reduce carbon emissions but also encourage long-term sustainability and will contribute towards green communication networks. This section presents the key resources relevant to ML-based network security.

3.1. Key performance indicators related to resource consumption

In terms of resource consumption, it is evident from the literature that most studies focus on energy/power, processing, bandwidth, memory, time, and human resources. Consequently, we considered these resources as the most common Key Performance Indicators (KPIs) for resource-efficient, ML-based network security solutions. Table 3 presents a summary of various ML-based network security approaches clustered in DDoS detection, DoS detection, anomaly detection, IoT network security, IDS, botnet detection, network traffic analyzer, and other network security applications. It highlights the type of network security addressed, the machine learning algorithms employed or compared, and the specific KPIs each solution covered. These solutions are discussed in detail throughout the latter part of our survey. Below, we briefly describe these KPIs.

3.1.1. Energy

ML algorithms learn from a large amount of data that requires computational power, memory, and, as a result, energy. Furthermore, the size, structure and parameters of a model also have impact on energy. Large and complex models have more parameters, require more data to train, and thus, consume higher amounts of energy. Additionally, different algorithms have different convergence and stability that affect the energy consumption while optimizing, and different optimization techniques also have different impacts on energy consumption [48]. The hardware, and software platforms, which enables an ML model to run, also requires energy. ML models for security require large and diverse datasets due to the complexity, rarity, and evolving nature of threats, as well as the need for context-rich, real-time analysis to minimize false positives. This increased data demand leads to higher energy consumption from intensive computation, storage, and data transfer, especially in large networks.

3.1.2. Processing

Processing power is a crucial resource for ML, which may involve the use of CPU, GPU, and TPU The requirement for processing power can vary based on the needs of applications and the nature of algorithms. A CPU performs general-purpose processing, sequential processing, data preparation, and other tasks. It is suitable for training simple models, small effective batch sizes, and models that are limited by the I/O or networking bandwidth of the host system. Additionally, for non-deep learning ML, the CPU is much more effective and cost-efficient. GPU consists of numerous simple processing units compared to the CPU and can handle parallel computing and perform rapid mathematical calculations. It is suitable for medium and large models with larger effective batch sizes. Finally, TPU is an AI accelerator particularly designed for TensorFlow and provides extremely fast execution of calculations [49]. Various factors, such as the size of the training and prediction dataset, may influence the amount of processing resources required by ML models for security. Larger datasets for anomaly detection will require GPUs, as they require higher computing power. Moreover, different ML algorithms have different complexity levels and require different levels of processing power: for instance, deep learning for packet inspection requires higher computing resources, such as GPU forms.

3.1.3. Memory

Memory is a crucial resource for ML, generally referring to the Random Access Memory (RAM). needed to temporarily store data that a computer processor can access quickly. The amounts of RAM required for security varies according to the application, which itself is dependent on the amount of data needed. For training ML algorithms, higher amounts of data is used, and thus higher RAM capacities are needed, whereas in inference comparatively smaller RAM would work. Furthermore, faster ML applications require stronger support from memory in terms of bandwidth, access speed, capacity, and reliability [50]. The long-term memory or storage is also a crucial to store the model after the completion of the training process. RAM is used during the training and inference processes to store the immediate or intermediate results and model parameters for quick access. Whereas storage is often used to store raw and processed training data and trained ML model parameters, which allows the trained model to be loaded and used later without re-training. For large models, besides memory, a significant amount of storage is also required. For example, LLMs, such as GPT 3, have over 175 billion parameters to store [50]. In security, some applications, such as anomaly detection, in networks with higher number of users and higher data rates would require higher memory and storage capacities, whereas, some applications, access control in lower data rate sensing devices would need lower memory and storage resources.

3.1.4. Time

Time is crucial in security, and is consumed in every step of the overall ML execution, including data collection, data cleaning, pre-processing, training, inference, decision implementation, and feedback loops. Most time is consumed during training, which varies based on different factors, such as the size of datasets, model complexity, and available computational resources. Moreover, data features also influence the training time, for instance, simpler features lead to faster training, while complex features require more time [51]. Additionally, time is a constraint in security, such as anomaly detection, DoS or DDoS prevention before they system gets exhausted, and therefore, extremely important. In a time-sensitive application, a model that can achieve 99.9% accuracy, but takes too long to make a classification decision, is essentially as ineffective as a model with 0% accuracy [52]. For example, availability is one of the key dimensions of security, as defined by the International Telecommunication Union (ITU) [13], and increased downtime due to security threats or prevention techniques is not acceptable. Therefore, time is a vital resource for ML, and its proper utilization is crucial in the security of communications networks.

3.1.5. Bandwidth

Generally, bandwidth, or network bandwidth, represents the maximum data transfer rate along a specific route [53]. It defines the number of bits, kilobits, megabits, or gigabits of data that can be sent per second. Recent communication technologies, such as optical fibre and terahertz frequencies, have enabled higher data rates in communications networks. However, bandwidth is a scarce resource for wireless networks, and the cost, for instance, in terms of frequency occupancy, goes up with the increase in bandwidth. When an ML-based solution is deployed for network security, it also uses the network bandwidth, for example, for gathering large volumes of data for real-time training, processing, and then deploying. Therefore, there is a need for bandwidth-efficient ML-based security algorithms that can be deployed in future communication networks without significantly increasing the network traffic.

3.1.6. Human

ML minimizes the need for human involvement in network security through automation, however humans are needed at various steps in the loops of ML system development, as discussed in [36,37]. These steps include data collection, feature selection, data transformation and pre-processing, model selection for model construction, parameter tuning, and assessing the quality of the result [54]. Many of these tasks require significant manual labour, often involving trial and error [55]. Human involvement is also crucial in the later stages of the ML lifecycle, such as deployment, retraining, and monitoring, as well as checking for false positives and false negatives. When various approaches are designed, developed, tested, and then released to the public without subsequent modifications, they may face challenges such as poor scalability, becoming outdated, performance degradation, and difficulty in evaluation when the deployment context changes. Additionally, due to the prevalent connectionist approach, ML models often lack logical reasoning capabilities and struggle to identify causal relationships [56]. In the domain of network security, the lack of human oversight can, thus, result in non-conformity of security policies, blockage of legitimate traffic, or even worse, for instance, misconfigurations of security devices, such as firewalls and security gateways. Hence, human resources are essential for guiding ML operations. It is pertinent to note that performing these tasks requires the expertise of field specialists, which can be very expensive.

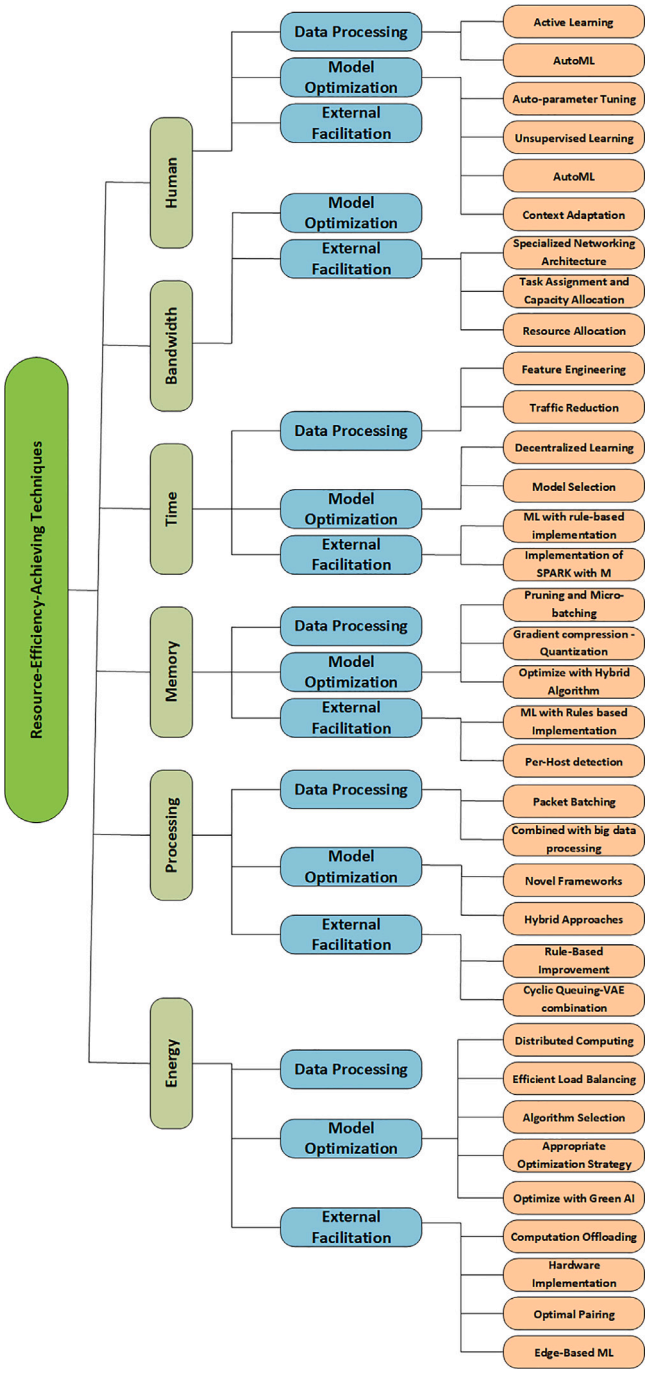


Fig. 2. Taxonomy of ML-based network security themes, with respect to selected KPIs, covered in this article.

3.2. Taxonomy

This article focuses on existing research that improves the above-mentioned KPIs for network security leveraging ML. Based on the identified literature, resource-efficiency-achieving techniques can be broadly discussed from three dimensions for the considered KPIs, which include the following: (i) data processing, (ii) model optimization, and (iii) external facilitation. Under each of these dimensions, there are different techniques for achieving resource efficiency. Fig. 2 presents

Table 3
Existing ML-based network security techniques that improve efficiency in terms of identified KPIs.

Ref	Security type	ML Algorithm	E	P	M	T	B	H
[57]	DDoS detection	Extreme Learning Machine (ELM)		✓	✓	✓		
[58]		VAE	✓	✓		✓		
[59]		LSTM						✓
[60]		CNN, RNN, LSTM, ERF, RFE		✓	✓	✓		
[61]		LR, SVM, DT, MLP, RF				✓		
[62]		LR, DT, KNN, RF, EBNB	✓		✓	✓		
[63]		REP Tree, Random Tree, RF, Decision Stump, PART		✓	✓			
[64]		DT, NB, SVM		✓	✓			
[65]		RF, Light GBM, XGB, Ada Boost Classifier (ABC)	✓	✓	✓	✓		
[66]	DoS detection	Custom	✓	✓	✓	✓		
[67]		KNN, DT, SVM				✓		
[68]		Lightweight variational Bayes algorithm	✓	✓		✓	✓	
[69]		LSTM		✓		✓	✓	
[70]		LSTM	✓	✓		✓		
[71]	Anomaly detection	FL, LR, FNN, 1D-CNN, VAE, Vanilla RNN, LSTM, GRU	✓	✓	✓	✓	✓	
[72]		FL		✓	✓	✓		
[73]		FL	✓	✓	✓		✓	
[74]		FL, Gated Recurrent Units (GRU)		✓		✓	✓	✓
[75]		KNN, DT, SVM, NB, RF	✓		✓	✓		
[76]		GA + Fuzzy logic						✓
[77]	IoT network security	FL, Deep Reinforcement Learning (DRL)				✓		
[78]		RR, ENR, Lasso, DT, RF, GBM, NB, and DL			✓	✓		
[79]		RF, DT, LogReg, AdaBoost, Ridge	✓					
[80]		FL, REDNN			✓	✓		
[81]		Custom						✓
[82]	IDS	SVM, RF, KNN			✓	✓		
[83]		DT, LSTM		✓		✓		
[84]		DT, RF, XGB, and KNN	✓			✓		
[85]		SVM and ANN				✓		
[86]		KNN, LR, DT, RF, NB, and ANN	✓					
[87]		Facebook's prophet	✓	✓		✓	✓	
[88]		Hybrid feature selection (Least Squares and SVM)		✓		✓		
[89]		ANN, SVM, RF, LDA, KNN, K-means, Mean-shift, DBSCAN				✓		
[90]		DT, RF, SVM, NB, CNN, KNN, ANN				✓		
[91]		MLP, DT		✓		✓		
[92]		RF, SVM, KNN		✓	✓	✓		
[93]		DRL				✓		
[94]		Custom				✓	✓	
[95]		RF, SVM, DQL, Bidirectional LSTM, CNN-BiLSTM					✓	
[96]		CNN1D, CNN2D, RNN, LSTM, DNN, Hybrid		✓	✓			
[97]		GPR, SVR, DT, Bagging, Boosting, KR, LR						✓
[98]		custom						✓
[99]		Hybrid (ACO-DNN)	✓	✓		✓		
[100]		MLP, kNN, XGB, RF, DT, SVM, LR, NB		✓	✓	✓		
[101]		custom		✓	✓	✓		
[102]		KNN, SVM, NN, Tree, and Ensemble						✓
[103]		RL	✓	✓		✓		
[104]		AutoML						✓
[105]		DT, NB, and kNN	✓	✓	✓			
[106]	Botnet detection	LR, RF, KNN, SVM, XGB, DNN		✓	✓	✓		
[107]		RL				✓		
[108]	Network traffic analyzer	AutoML						✓
[109]		LGBM, RF, ETC, DT, GB, kNN, LDA, RC, ABC, QDA						✓
[110]	Other network security applications	DNN	✓					
[111]		RF, LR, KNN, GNB, DT, SVM (RBF), MLP, GB, XGB			✓	✓		
[112]		SVM, MLP	✓			✓	✓	
[113]		PCA, LDA, and SVM				✓	✓	
[114]		FL (MLP)	✓			✓		
[115]		RL, NN				✓		✓
[116]		KNN, MLP, RF, LightGBM, Custom						✓
[117]		AE, K-Means						✓
[118]		SVM, MLP, DT, and RF					✓	
[12]		DNN	✓					

Bandwidth: B, Energy: E, Memory: M, Processing: P, Time: T, Human: H.

the taxonomy within these three dimensions for each of the KPIs. It is evident that different techniques are used to improve the resource efficiency of ML models in securing communications networks. A brief description of each dimension for the outlined KPIs is provided below.

3.2.1. Data processing

A common tendency we observed in various network security approaches is the reduction of resource consumption by efficiently processing input data before feeding it into ML algorithms. Generally, high-dimensional data includes a lot of noise and irrelevant features,

making it essential to pre-process the data and select the right features before training a model. This ensures the model focuses on the most important information [119]. In this way, both the dimensionality of the data and the required resources to process the data are reduced. Different techniques are used to process data, resulting in reduced resource consumption. The techniques we observed during our study include feature engineering, feature selection, feature extraction, traffic reduction, micro-batching, packet batching, and utilizing techniques commonly used in big data processing.

3.2.2. Model optimization

Model optimization is a crucial part of ML operation towards the success of ML models. It involves repeated modification of model parameters to minimize or maximize a predefined objective function, such as loss function in supervised learning [120]. The main goal of the optimization technique is to discover the parameters that best align with the given data and excel when applied to new data, thereby improving prediction, precision, and overall performance [121]. We have studied different ML solutions, algorithms, and frameworks that are used to reduce resource consumption using model optimization. These techniques involve model compression techniques, like pruning and quantization, algorithm selection, decentralized learning, optimization with hybrid algorithms, distributed computing, unsupervised learning, auto-parameter tuning, context adaptation, dynamic evolving of NN, efficient load balancing, and appropriate optimization strategy selection.

3.2.3. External facilitation

Various external facilitators or techniques, not typically involved in regular ML operations, are utilized in different ML-based network security solutions to reduce resource consumption. For instance, hardware accelerators, computation offloading, processing ML at the edge, or end-user nodes, and different networking architectures (e.g., software-defined networking (SDN)), task assignment and capacity allocation, resource allocation, implementation with the rule-based methods, utilization of specialized computing platform, per-host detection, combining with queuing algorithm, specialized hardware or software usage, and optimal pairing techniques are used to improve resource efficiency. In our study, we found that different ML-based network security solutions leveraged these external facilitators to improve performance and reduce resource consumption.

In the following section, we discuss different ML techniques used for network security according to these KPIs and the taxonomy outlined in this section.

4. Resource consumption of ML in communications networks security

In this section, we study and analyse the resource efficiency of ML techniques used in network security. We have categorized the existing research in terms of the outlined KPIs, such as energy, time or latency, memory, and processing or computing resources. We have discussed each of these KPIs with the outlined taxonomy, such as model optimization, external facilitation, and data processing. Within these three main categories, there are various approaches, such as computation offloading, load balancing, etc., for improving resource efficiency. Therefore, existing research is discussed with respect to these approaches in the following subsections. Below, we discuss ML-based security approaches that are energy efficient.

4.1. Energy

In this subsection, we discuss various efforts by researchers to reduce the energy consumption of ML-based network security through data processing, model optimization, and external facilitators. Table 4 presents the performance of the ML algorithms, the datasets used, and other relevant information for these network security approaches.

4.1.1. Data processing

Julio et al. [79] explored the relationship between dataset size, energy consumption, and the accuracy of ML models in IoT anomaly detection. Based on empirical analysis with various ML models and multiple datasets, the authors analysed the impact of instances and the number of features on the energy-accuracy trade-off. The findings indicate that it is feasible to attain 98.5% of the highest accuracy using just 10% of the dataset's samples, which also leads to reduced energy consumption. However, reducing the number of features can lower energy consumption but may negatively impact accuracy. Pasikhani et al. [103] reduced the energy consumption of their Adversarial RL-based IDS by optimizing data processing, minimizing communication overhead, and efficient ML method. The RL-based IDS only preserves the operation performance-critical data and discards the less critical data to deal with the resource-constrained nature of the nodes. Instead of actively looking for possible intrusions of various sizes, the system uses a passive approach. Where IDS agents monitor network traffic in promiscuous mode without actively interfering, resulting avoidance of generating additional communication overhead. This process improves the ability of the Defender Agent to detect the most sophisticated intrusions. The authors used a wide range of routing attacks to evaluate the scheme in both a Black-box and a Grey-box environment and found it effective, reliable, general-purpose, and resource-efficient (computational, energy resources, and latency) for IDS.

4.1.2. Model optimization

The model optimization approach comprises novel algorithms, frameworks, and methodologies that impact ML from an algorithmic perspective.

Distributed Processing: Distributed processing is the process of using multiple computing resources or computers connected over a network, working together to solve a computational problem. The process involves distributing workloads to multiple nodes to work collaboratively to achieve a common goal, which has many advantages over a centralized operation used for processing large amounts of data. He et al. [68] utilized the benefits of distributed processing to reduce communication resources and energy in their Distributed Variational Bayes-based in-network security for DDoS detection. Here, the algorithms are directly implemented into the switches to detect anomalous traffic. A centralized platform synchronizes parameters to compensate for the limited training data at each switch, enabling collaborative learning. The algorithm has two parts: the first part spreads information throughout the network, while the second part uses local data and information from neighbouring nodes to gradually update the estimator. This presented algorithm reduces communication dimensions and computational load compared to centralized processing, thereby reducing energy consumption and computational resources.

Optimize with Green AI: Green AI refers to the research that produces innovative results while considering computational costs, thereby encouraging the reduction of resource usage. In contrast, red AI led to a significant increase in the usage of computational power and resources, and also increased carbon emissions. Green AI, however, advocates for methods that balance performance and efficiency effectively [12]. Alberto and Amit [122] employed green AI techniques to optimize the size and complexity of ML models, resulting in reduced energy consumption for their cyber threat identification approach for a cloud-based SDN controller. The advanced optimization technique was developed by combining 11 different strategies. These strategies aim to reduce the energy consumption of deep learning-based attack detection models in the TeraFlowSDN centralized attack detector component. The result shows that energy consumption can be reduced by up to 83.30% with only a slight performance decrease of 0.08% accuracy.

Efficient Load Balancing: The concept of load balancing was introduced during the initial implementation of distributed computing. It refers to the even distribution of the loads to a set of servers with the goal of throughput maximization, response time reduction,

Table 4
ML techniques used for security that improve energy efficiency.

Ref	ML Algorithms*	Datasets	Energy efficiency achieved by	Validation techniques	Acc.	Pr.	Re.	FS
[70]	LSTM	–	Computation offloading	Cross-validation	94	93	94	90
[84]	RF	CICIDS2017		Cross-validation	99.90	99.90	99.90	99.90
[86]	DT	Custom	Model Selection	Comparative analysis	99	99	96	98
[71]	FL (LR)	KDD 99		Holdout	96	61	93	–
		NSL-KDD			92	64	95	
		CIDDS-001			91	70	91	
[112]	MLP	Custom	Efficient load distribution	–	84–89	–	–	–
[62]	NB	KDD 99 CUP	Edge processing	–	94.0	95.0	91.0	96.0
		ISCX			93.0	97.0	93.0	94.0
		BOT-IOT			95.0	97.0	95.0	96.0
		KITSUNE			92.9	92.9	94.0	92.0
		KITSUNE+ GD			94.9	93.3	99.9	96.4
[105]	DT	Custom	Hardware Implementation	Stratified Sampling	99.14	–	–	–
[99]	Hybrid (ACO+ DNN)	KDD Cup 99		Cross-validation	99.5	99.6	96.6	99.8
		NSL-KDD			92.9	93	92.9	93
[114]	FL (MLP)	MNIST	Optimal pairing	Comparative analysis	79	–	–	–
[68]	Hybrid	KDD 99	Distributed processing	Cross-validation	86.52	94.2	–	–
[110]	DNN (FCNN)	Crypto dataset (Custom)	Energy optimization strategy	Cross-validation	99.5	–	–	–
[79]	AdaBoost	IoTID20	Training with fewer instances and features	Cross-validation	100	–	–	–
		BoTNeTIoT-L01			98.86			
		X-IIoTID			99.18			
		IoT-DNL			80.04			
[12]	DNN	Custom	Optimize with green AI	–	99.54	99.99	99.54	99.54

ML Algorithms*: Best Performing ML Algorithms, Accuracy: Acc., Precision: Pr., Recall: Re, F1-score: FS.

and increment of the resilience of the system to faults by avoiding overloading the systems [123]. Fig. 3 shows a conceptual load balancing scenario, where a load balancer distributes the requests from two clients to two distinct servers. The concept of load balancing was utilized in [112] to develop a Load-Aware, energy-efficient, and Secure Weighted Clustering Algorithm (LESWC), which is a data-driven ML-based approach. It is an extension of EE-WCA [124] with a centralized IDS and some modifications in the data processing algorithm. Here, the ground-level sensor nodes are deployed using the EE-WCA algorithm. The Base Station (BS) consists of an MLP algorithm to classify traffic data and recognize low traffic and secure routes, and detect malicious activity. This smart BS can make smart decisions and alerts about various network attacks. Additionally, it can decide on the arrival of sudden workload, and hence, is effective in various mission-critical situations. Moreover, since the algorithm can predict the load one step ahead and maintain network lifetime by uniformly distributing energy consumption among all the network nodes equally, it is more efficient and reliable in terms of buffer, computational overhead, and energy consumption.

Model Selection: By comparing different ML algorithms in a specific context, valuable insights can be gained regarding their performance variability, energy efficiency, scalability, and execution time. These insights can be utilized when deploying an ML-based solution in an energy-constrained scenario. Tekin et al. [86] compared the energy consumption of various on-device ML algorithms for intrusion detection in three different training methods (cloud, edge, and On-device) and two inference approaches. Energy consumption is measured by using both execution time and power measuring tools, while considering that energy consumption is directly proportional to the execution time. The experiment considered six ML algorithms, and they are KNN, LR, DT, RF, NB, and ANN, with three different dataset sizes. The result shows that the training time for all ML algorithms increases as the dataset size increases. The training time increases drastically for ANN. NB takes significantly less training time across all the execution platforms, resulting in lower energy consumption. However, NB's accuracy was 97%, which was the lowest. DT and KNN also require relatively less training time while achieving high accuracies like RF and ANN across all the platforms. The cloud can significantly reduce execution time for the algorithms that support multiple cores, such as RF, but this benefit is not observed in other algorithms. The KNN takes much higher inference time than the training time, hence it is not suitable for on-device deployment, even though the training is performed in edge or cloud. The power monitoring tool confirmed that the DT algorithm maintains high accuracy while consuming much less power than the other algorithms. Jithish et al. [71] showed that the FL-based model can perform efficiently in terms of memory, bandwidth, power consumption, and processing, and can be implemented in resource-constrained environments in anomaly detection. The authors developed seven FL-based models, used three anomaly detection data sets, and compared the models with the centralized ML models. The algorithms used to develop those models were LR, Feedforward Neural Network, Autoencoder (AE) Binary Classifier, 1D-CNN Binary Classifier, LSTM Binary Classifier, Vanilla RNN Classifier, and GRU Binary Classifier. The result shows that for FL training, LR and GRU binary classifiers consume fewer CPU resources, whereas the 1D-CNN requires intensive computational resources. According to the result, compared to centralized ML-based solutions, this FL-based anomaly detection used only 10%–31% of memory, 5.56–88.78% CPU, 0.33–2.34 W power, and has 48.8–1191 kbps communication overhead, hence uses less bandwidth.

Appropriate Optimization Strategy Selection: Different optimization techniques, such as pruning and quantization, can reduce the

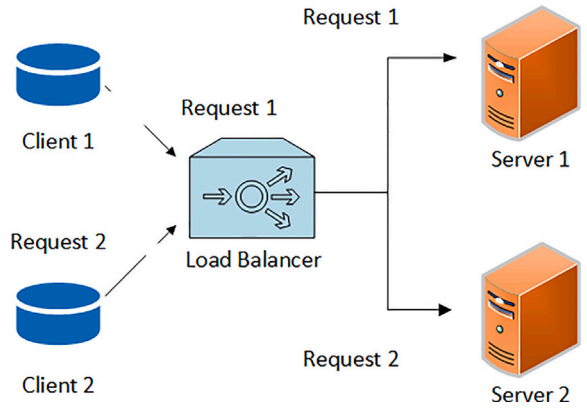


Fig. 3. A conceptual diagram of load balancing.

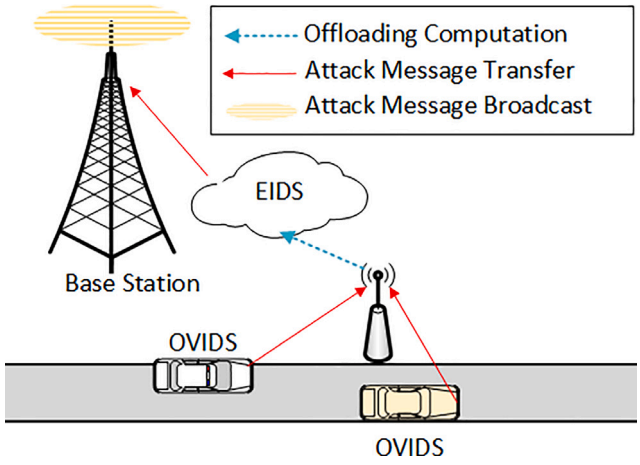


Fig. 4. Computation offloading of vehicle security system to computation at edge.

energy consumption of DNN. However, identifying the best combinations of those optimization techniques can reduce energy consumption significantly. Amit et al. [110] introduced a comprehensive methodological framework to assess the energy consumption and resource utilization of DNN-based systems in production settings. It enables the evaluation and comparison of various novel optimization techniques, offering a detailed understanding of the trade-offs between energy efficiency, resource usage, and performance. Additionally, the framework introduces the concept of an "optimization profile" to manage the balance between energy savings and model performance, which quantitatively represents the scenarios where either energy efficiency or performance may be prioritized. Furthermore, some of the optimization techniques examined can be combined, as they are not mutually exclusive, to further decrease the model's energy consumption. The framework was validated on a DNN-based cyber threat detector and found that it can reduce energy consumption during inference by up to 82% with a little accuracy loss.

4.1.3. External facilitation

Various external facilitators can aid in achieving energy efficiency for ML-based solutions, such as computation offloading, hardware implementation, optimal pairing, and edge-based approaches.

Computation Offloading: The energy consumption of a device can be reduced by computation offloading. This method entails transferring a certain portion of the computational task of a resource-constrained device to a more powerful computing system. This approach has many benefits. However, it could negatively affect the energy consumption of devices with limited power if channel conditions deteriorate during the offloading process [125]. Mirzaee et al. [84] proposed a two-layer IDS system, where the attack is detected through collaboration between the autonomous vehicle's own IDS (OVIDS) and the IDS from the mobile edge (DISD). The model can reduce energy consumption and communication latency. Here, the data passed through the OVIDS are aggregated and classified into normal and abnormal batches. The abnormal data are then sent to the edge intrusion detection system (EIDS) via Roadside Units (RSUs) for further investigation. A generic demonstration of this computation offloading is shown in Fig. 4. The authors used the CICIDS2017 dataset and implemented the model with different ML algorithms, such as RF, DT, XGB, and KNN, in the vehicle's detector, where RF performs the best in the whole process. The distributed detection computation in vehicles and the infrastructure servers limits the data size to only the abnormal parts, which reduces both energy and latency. According to the result, the IDS can detect attacks with a 99.90% accuracy, while reducing the latency and vehicular energy consumption up to 80%. Eric et al. [70] proposed a Multi-Trust DoS Attack Detection System, where parameters captured from IIoT devices are offloaded to a deep neural network model created with LSTM, hosted at the edge (MEC). This approach overcomes the resource constraints of IoT devices that limit on-device ML model training. To address the high latency associated with DDoS attacks, the approach employs a hold-and-check filter on the IIoT devices. The results show that M-TADS performs well in terms of energy consumption, throughput, accuracy, and packet delay.

Hardware Implementation: Hardware Implementation (HW) refers to a circuit designed with a dedicated purpose, while Software Implementation (SW) refers to a solution running on a general-purpose hardware or computing platform. Viegas et al. [105] demonstrated that the energy consumption could be reduced significantly by HW of network security techniques and algorithms than the SW. The authors presented three techniques to improve the energy efficiency of anomaly-based intrusion detection engines for embedded and battery-powered devices. The first technique is a feature extraction algorithm named the hash-based extractor. This approach is suitable for HW implementation and demands low computational resources. Unlike traditional feature extraction, this technique does not need a series of lookup operations and indirections, and it allows a simple resize of the time window. The result shows that the hash-based extractor consumes only 0.33% energy, 189 times faster than the compared traditional approach. Additionally, it is 8.7 times faster and consumes only 22% energy compared to a commercial solution. The second one is a feature selection method that can improve the energy efficiency and accuracy of the model. Here, three scenarios were considered: no Feature Selected as the baseline, single objective where only accuracy was the goal and dual objective where accuracy and energy consumption reduction were the goals. Results showed that the dual objective feature can save 9.3% of energy compared to the single objective feature selection while compromising 0.90% of accuracy. Finally, the hardware implementation of the ML classifiers with the feature extractor engine was examined using DT, NB, and KNN classifiers. The comparison was made with no selection, single objective, and dual objective feature selection approaches. The result shows that a dual objective with DT is the most energy-efficient classifier that can classify one packet with only 15pJ, which is only 0.03% of the corresponding SW implementation.

Dynamic voltage and frequency scaling (DVFS) is a widely used power management method that lowers a processor's clock frequency, enabling a decrease in voltage supply. This approach can reduce significant levels of energy consumption, especially in memory-bound tasks [126]. Jitendra et al. [99] adopted the DVFS mechanism to minimize the energy consumption in their cloud computing-based IDS that uses high-dimensional data. Modern CPUs prefer using the DVFS method to adjust their frequency, addressing energy efficiency concerns dynamically. Typically, CPU utilization aligns with the system's overall

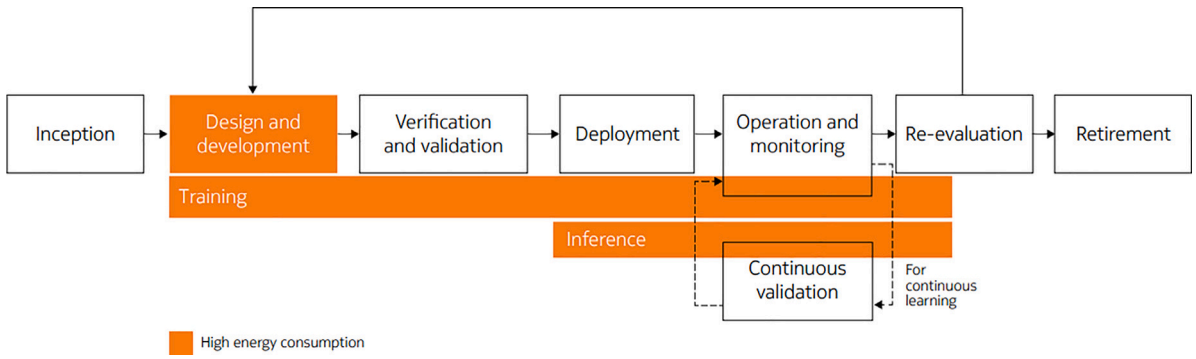


Fig. 5. AI system life cycle with high energy consumption processes marked with colour [11]. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

load. The authors used Ant colony optimization (ACO) for data dimensionality reduction. The work aimed to develop a balance between energy consumption and operational time of various modes of virtual machines (VMs) or hosts. The techniques were compared with ACO and PCA-based NB models, and the proposed ACO-DNN outperformed these models in terms of performance, energy consumption, and detection time.

Optimal Pairing: In general, the trained wireless FL models are sent over the radio channel, and attackers can take advantage of the open access nature of the radio network and collect the FL client's (FCs) information from a malicious node and decode that information by brute force. To prevent this eavesdropping attack on the FCs, an artificial jamming-based [127] channel-sharing approach is used, which is a physical layer-based security that measures the secrecy capacity that cannot be eavesdropped on. Some friendly sensor nodes (SN) send jamming signals that interfere with the eavesdropper's signal. However, the implementation of this jamming technique increases energy consumption. Developing an optimal pairing strategy for these FCs-SNs can minimize system cost and latency. Wang et al. [114] propose a channel-sharing scheme with artificial jamming where each federated client (FC) lets a nearby sensor node (SN) use its radio channel during local training, and in return that SN transmits a jamming signal while the FC uploads its model, boosting secrecy throughput via physical-layer security. The authors formulate a bicriteria cost that combines the total energy of all FC–SN pairs and the maximum end-to-end latency. A decomposition method optimizes power and time for every given pair, and an efficient Algorithm-CMP finds the globally optimal pairing with energy-latency trade-off: it first minimizes energy, then iteratively adjusts the matching to drive down the bottleneck latency until no further reduction in the weighted cost is possible. Finally, the performance of the model was compared with various benchmark solutions, where it outperformed them in terms of latency and energy.

Edge-based ML approach: Edge-based ML can be employed to handle various types of traffic analysis, perform dedicated feature selection, and significantly improve the packet processing rate. Pradeepkumar et al. [62] proposed a computation and storage-efficient, edge-based IDS capable of detecting DDoS attacks from specific sources in 6TiSCH networks. This approach captures traffic using Wireshark, extracts features from the traffic using PCA, and classifies them with a Gaussian NB classifier. During the training of the edge-based ML approach, only a single instance is kept in memory at any given time. This parameter is employed to improve response time while balancing detection performance. When compared with other approaches in the literature, this method achieved 98.7% accuracy, used 35,834 bytes of memory, consumed 85,916 mJ of energy, and had a response time ranging from 24.2 to 68.9 s, showing significant improvements over other methods.

4.1.4. Key takeaways

Energy is a critical KPI for machine learning (ML) in network security. ML models can consume a significant amount of energy, particularly during the training and inference phases. As illustrated inFig. 5, the majority of energy consumption occurs during these two stages, followed by design and development to a lesser extent. Other stages, such as inception and retirement, typically require less energy [11]. This high energy demand is primarily due to the processing of large datasets, complex model architectures, and the need for continuous learning. Three main strategies have been identified for reducing energy consumption: data processing, model optimization, and external facilitation. Our observations indicate that most efforts have focused on model optimization and external facilitation. Notably, many authors employed overlapping strategies. For example, authors who used external facilitators, such as hardware acceleration, computation offloading, and edge-based ML, incorporated data processing techniques like dimensionality reduction and feature extraction. Similarly, authors who applied model optimization techniques, including distributed processing, green AI, model selection, efficient load balancing, and optimization strategy selection, relied on data processing methods. These include using smaller datasets, optimizing batch sizes, or applying feature selection, feature extraction, and label encoding to reduce energy and latency. From this perspective, data processing emerges as the most common and impactful strategy, widely used across various network security applications to enhance performance and reduce energy usage. In addition to reducing energy consumption, many of these approaches also contribute to lowering latency and CPU usage. However, all three strategies may lead to some degradation in accuracy. Finally, the external facilitation technique known as optimal pairing is applicable only to distributed ML systems, such as federated learning (FL), particularly when jamming techniques are employed to enhance security.

4.2. Processing

This subsection discusses various data handling, model optimization, and external facilitation techniques that help reduce resource consumption in ML-based network security. Table 5 presents the performance of the ML algorithms for these solutions, along with other important information.

4.2.1. Data processing

Processing power efficiency can be achieved by employing various efficient data processing techniques, such as batch processing and big data processing methods.

Packet Batching and Pre-trained Model usage:

Packet aggregation is the way of combining multiple packets into a single transmission unit to minimize the overhead of each transmission in a packet-based communication network [128]. These packets can be aggregated in different batch sizes, and the appropriate batch size can reduce processing costs. Gandhi et al. [106] focused on data preprocessing to reduce the amount of data fed to the ML classifier, thereby

Table 5
ML-based security techniques that improve processing (CPU and GPU) efficiency.

Ref	ML Algorithms*	Datasets	Processing efficiency achieved by	Validation techniques	Acc.	Pr.	Re.	FS
[65]	RFs	Slowloris + CICDDoS2019	Combining ML decisions with binary logic (AND gate)	Cross-validation	99.12	99.95	99.96	99.80
[113]	SVM	Custom	Dimensionality reduction	Cross-validation	99	–	–	–
[83]	DT	UNSW-NB15	Snort rule generation, feature extraction	Cross-validation	99.79	99.73	99.86	99.8
[103]	RL	Custom	Feature selection, ARL, feature extraction, feature engineering	Empirical: black-box grey-box	96.3 92.2	96.3 92.2	96.29 92.22	96.29 92.22
[69]	LSTM	CICIDS2017	Period-wise detection, packet-wise detection, Feature selection	Cross-validation	–	99 99.4	99 100	99 99.7
[58]	VAE	A5M B5M	VAE-Cyclic-Queuing combination	Holdout	–	–	–	–
[106]	DNN	Custom	Packet batching	–	85	–	–	85
[64] [100]	DT MLP, kNN, XGB, RF, DT	Custom Custom	Novel framework	Holdout Cross-validation	98.1 –	– –	– –	– 100
[60] [101]	ERF&RFE Custom	CICIDS-2017 UNSW-NB15 VeReMi	Hybrid approach	Holdout –	99.96 98 96	99.77 –	97.37 –	98.53 –

ML Algorithms*: Best Performing ML Algorithms, Accuracy: Acc., Precision: Pr., Recall: Re, F1-score: FS.

reducing CPU and memory usage. In doing so, the authors utilized a packet-batching technique. The authors studied the correlation of network traffic and the burst of traffic classification to profile the system resource consumption for botnet detection and remarked that packet aggregation can reduce CPU, time, and memory consumption. Fig. 6 shows a conceptual representation of packet batching, where the arriving packets are aggregated as per batch size. However, it slightly reduces the accuracy, but it significantly improves the F1-score. The ML algorithms used in this study were LR, RF, KNN, SMV, XGB, and DNN, while the models were pre-trained to avoid high memory usage for training. According to the authors, well-known methods are not suitable for on-device training and testing because of the high computational resource demand. Additionally, it is worthwhile to explore online training for resource-constrained applications. For the experiment, the packets were aggregated in three different sizes: 1000, 5000, and 10,000 packets. The experiment result shows that, while the batch size is smaller (reduced to 10,000 to 1000), the performance degrades and consumes more computational resources. Memory consumption decreases drastically with the increase of batch size. The training time also decreases with the increment of aggregation size, while SVM shows the sharpest decrease, which is 20x. Finally, the peak CPU usage decreases slightly with the increase of batch size for most of the models, while the neural network showed an exception of a 5x decrease in peak %CPU usage. Finally, XGB consumed significantly less memory than the other algorithms during model training for all the batch sizes. LR, KNN, and XGB have lower training time, and LR had the lower peak CPU usage. The authors recommended using pre-training for the models that have a higher memory usage footprint.

Combined with big data processing Big data analytics depends on tools for data visualization, sophisticated data processing techniques, and ML algorithms. These elements help organizations identify patterns, trends, and anomalies in large and varied datasets that might otherwise remain hidden. The potential benefits are numerous, including cost reduction, improved decision-making, increased revenue, and customer experience [129]. Integrating big data processing techniques with ML-based network security can meet online detection with improved accuracy, good scalability, and latency. Shi et al. [69] presented DeepDDoS, which is a deep learning method based on both period-wise detection and packet-wise detection. The two-step DDoS detection framework can reduce computational overhead and delay. The framework has four components: network traffic collector, message delivery through Kafka, Spark streaming statistic, and LSTM components. Here, Spark can continuously monitor and analyse network data whenever it is transmitted. Hence, it can be very useful to detect DDoS attacks. There are two LSTM-based models: DeepDDoS-1, which is period-wise detection; and DeepDDoS-2, which is packet-wise detection. The period-wise detection reduces the computational overhead and delay. The data packets are then grouped by five tuples and used as input in DeepDDoS-2. The packet-wise detection can use very few packets to detect an attack. For instance, even though there may be a thousand packets, this method can detect an attack by only relying on five consecutive packets with satisfactory performance and high accuracy.

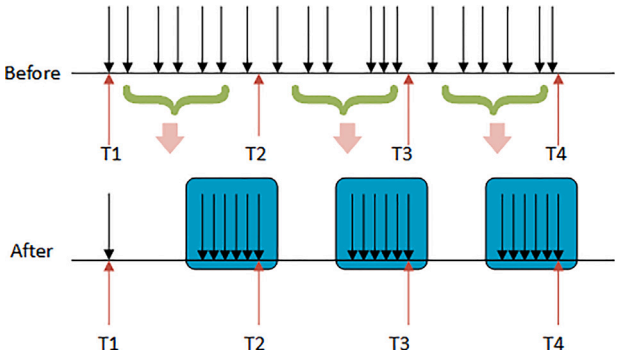


Fig. 6. The arriving packets are aggregating in batches.

4.2.2. Model optimization

This section highlights various methods used to enhance processing efficiency through different optimization strategies, such as developing novel frameworks and hybrid approaches.

Novel Frameworks: Jalal. et al. [64] presented a DDoS detection framework for an SDN-WISE IoT controller, which can significantly reduce CPU and memory usage. The framework consists of three modules: (i) a data plane module, which includes an SD-IoT network, Sensor

OpenFlow Switch (SOFS), and IoT devices, managing incoming IoT traffic and acting as a gateway; (ii) an IoT controller module, which uses an adjusted SDN-WISE to manage traffic and direct switches; and (iii) an ML-based DDoS detection module, which classifies IoT node traffic to detect DDoS packets. The results show that the framework can demonstrate accuracy rates of 97.4% for NB, 96.1% for SVM, and 98.1% for DT, while the system uses only 30% of CPU and memory, conserving 70% of CPU and memory to handle SD-IoT network traffic. It processes an average throughput of 48 packets per second, achieving an overall accuracy of 97.2%. Gustavo. et al. [100] presented a five-layer end-to-end framework for ML, named AB-TRAP which can achieve very high detection accuracy with a very minimal CPU and RAM usage. The framework consists of five steps, such as generating an attack dataset, creating a dataset, training ML models, implementing the models, and evaluating the performance of the deployed models. The framework was tested in TCP port scanning attacks in both LAN and the internet. In the LAN, it achieved 96% F1-score, and 99% ROC curve using a DT with minimal CPU and RAM. In the internet scenario, it was tested with eight ML algorithms with an average ROC 98% and average F1-score 95% with an average overhead of 1.4% CPU and 3.3% of RAM usage.

Rani et al. [65] proposed a framework to identify and prevent DoS and DDoS attacks in a D2D communication environment, where the framework's capability and lifetime are mainly associated with the CPU, battery, and memory of D2D devices. Initially, the authors examined GBM, XGBoost, AdaBoost ML, and RF models in terms of hyperparameters and feature optimization through extensive emulation with CICDDoS2019 and a D2D-specific Slowloris attacks dataset. The results show that RF can achieve greater accuracy with both datasets. Based on the result, a new approach is introduced to evaluate a joint result of two or more binary classification RFs, using a simple binary logic (using AND gate) instead of making a complicated structure of RF for both. Here, one of the RFs was trained with SYN and DDoS, and another RF was trained with SlowLoris attacks. This presented approach was found superior to other compared approaches in terms of detection and prevention time, memory, and processing resource requirements, and device battery consumption without compromising detection accuracy.

Hybrid Approaches: Tanut et al. [60] presented a hybrid algorithm by combining Recursive Feature Elimination (RFE) and Ensemble Random Forest (ERF), which conserves more computing resources compared to methods that rely on neural networks. The integration of these two techniques allows the algorithm to take advantage of the strengths of both ensemble learning and feature selection. This helps achieve better performance in terms of accuracy, precision, testing time, and CPU usage for DDoS attack detection. The results show that the presented algorithm (ERF&RFE) can achieve 99.96% accuracy, with only 4.6% CPU usage, and 15.51 and 0.0039 min for training and testing time, respectively. In comparison, RNN+LSTM had 98.75% accuracy, required 8.48% CPU, and took 21.54 and 0.028 min. For CNN+CNN, it was 99.45% accuracy, 10.036% CPU, and 78.52 and 0.13 min. Jie. et al. [101] presented LH-IDS, a hybrid lightweight IDS for Vehicular Ad hoc Networks (VANETs), which has a very low impact on CPU and memory usage. The hybrid approach was developed based on Differential Privacy (DP) and unsupervised ML. Here, the unsupervised ML detects anomalous network behaviour, including unknown attacks, while DP preserves the privacy of training data. The result shows that this approach can detect anomalous behaviour with 98%, 96% with UNSW-NB15 and VeReMi datasets. Additionally, it uses approximately 6% of CPU resources and 1% of memory, thus imposing minimal overhead on VANETs.

4.2.3. External facilitation

This segment presents the solutions wherein processing power efficiency is attained through the deployment of external facilitators such as rule-based implementation and cyclic-queuing with ML.

Rule-Based Improvement: Among the various challenges involved in integrating IDS into the safety-critical embedded system, traceability and efficiency are crucial. Traceability provides transparency and accountability in a safety-critical system by allowing the tracking of decisions and data from their source. However, efficient resource utilization is important because the safety-critical embedded systems are resource-constrained. Lucas. Et. al.[83] proposed an approach where ML is used to automatically generate rules for a rule-based IDS, enabling traceable and efficient intrusion detection. This addresses the challenge of deploying an IDS in safety-critical systems, as rule-based systems are both verifiable in behaviour and consume fewer resources. It combines the rule generation with the extension of the DL dataset. Here, DT is used to generate some traceable Snort rules, which can be used as a basis for the safety-critical rule set, instead of continuously running ML for detection purposes. An LSTM is used to extend the dataset for the rule generator to address the lack of data. It has three main components: (i) Rule-Based detector, which has very high traceability of its decision, superior computation performance, and reduced complexity; (ii) Rule Generator, which generates traceable and reliable rules and is periodically retrained; and (iii) anomaly detector, which extends the dataset for the rule generator. Finally, a traffic logger records all the traffic within the IDS operating environment. The result shows that the IDS is practical and effective for deriving IDS rules for safety-critical systems. However, using DL-based techniques to generate datasets can lead to misclassification if there is any error in the synthetically extended dataset.

Cyclic-Queuing-VAE combination: Cyclic queuing is a promising technique for DDoS detection in resource-limited network edge, where the attack flows are identified by repeated reconfiguration of queue mapping. However, the continuous reconfiguration during normal flow is computationally intensive and causes power consumption. To address this challenge, Yaegashi et al. [58] proposed two-stage DDoS mitigation techniques for the network edge node, which employs both VAE and Cyclic Queuing together for efficient and fast DDoS detection. The first stage is the transition stage from normal to attack state, where the anomaly is detected with VAE. If the anomaly is detected, it alters the state to the anti-attack state. In the second stage, the attack is detected by cyclically re-configuring the queuing map. It improves the detection speed by narrowing down the suspected flow and utilizing the history of queuing size around the stage transition. The simulation result found the performance of the technique consistent with the theoretical value, and it can detect attack flow successfully.

4.2.4. Key takeaways

We observed that most approaches that focused on CPU efficiency may also consider memory and time efficiency. For example, data processing techniques such as packet batching, when combined with pre-trained models, can lead to significant savings in CPU, memory, and processing time, though with a slight compromise in accuracy. Additionally, big data frameworks like Spark and Kafka can help reduce CPU usage and processing time. Model optimization using novel frameworks and hybrid approaches can significantly reduce CPU and memory consumption while maintaining high accuracy. Moreover, we observed that the model optimization category received more focus than data processing and external facilitation in terms of CPU efficiency. However, the authors used model optimization and external facilitation also used data processing techniques. For instance, several presented novel frameworks incorporated data processing methods such as normalization, handling missing data, feature scaling, and encoding categorical variables. The presented hybrid approaches employed feature selection and other preprocessing techniques. Finally, data pre-processing is also involved in the external facilitation: rule-based improvement. Therefore, data processing emerges as the most commonly used technique for improving CPU efficiency across all categories.

Table 6
ML-based security techniques that improve memory efficiency.

Ref	ML Algorithms*	Datasets	Memory efficiency achieved by	Validation techniques	Acc.	Pr.	Re.	FS
[63]	PART	CICIDS2017	Feature selection	Holdout	99.7	–	–	–
[92]	RF	Custom		Holdout	93.09	–	–	–
[66]	Custom	–	Integration with rules implementation	Empirical analysis	100	–	–	–
[82]	RF	Bot-IoT	Per-host detection	Cross-validations	99.8	–	–	–
[106]	XGB	Custom	Packet batching	–	85	–	–	85
[96]	Hybrid	CICIoT2023	Gradient compression and adaptive secure aggregation strategies	–	91.35	–	–	–
[80]	DNN	Danmini Doorbell Ecobee Thermostat Ennio Doorbell Philips B120N10 PT-737E PT-838 SNH-1011 XCS-1002 XCS-1003 Kitsune Wustl	Optimization: micro-batching, pruning, parameter regularization, weight averaging	Hold-out	95.11 93.36 88.94 84.08 88.07 92.52 86.06 94.65 97.73 84.09 94.26	–	–	–
[111]	RF	Modbus TCP/IP MQTT	Optimizing ML by blending with a modified hybrid bat algorithm (HBA) aggregation strategies	Cross-validation	95 90	95 92	–	95 90
[78]	DT GBM RF	DS2OS	Model Selection	Holdout	99.99 100 99.4	–	–	–

ML Algorithms*: Best Performing ML Algorithms, Accuracy: Acc., Precision: Pr., Recall: Re, F1-score: FS.

4.3. Memory

Different data processing, model optimization, and external facilitation techniques have been used to reduce memory consumption in various ML-based network security solutions. In this subsection, we include some of these solutions, with Table 6 showing their performance and other important information.

4.3.1. Data processing

Feature selection involves identifying a subset of features from the original set that are considered "relevant". The importance of these features is determined by the system's goals [130]. Features can be categorized based on their relevance in three criteria. The irrelevant features are excluded from the original set. Weakly relevant features are those whose addition to a subset does not improve performance. Strongly relevant features are crucial, as their removal significantly degrades performance. Feature selection methods either assign weights to individual features or evaluate subsets using a search strategy [131]. Eq. (1) shows a visual representation of feature selection, where X_1, X_2, …, X_N, are the initial feature set. After the feature selection process, a smaller feature set is obtained with the selected features X_{i1}, X_{i2}, …, X_{im}.

$$\begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_N \end{bmatrix} \xrightarrow{\text{feature selection}} \begin{bmatrix} X_{i1} \\ X_{i2} \\ \vdots \\ X_{im} \end{bmatrix} \tag{1}$$

To improve the DDoS detection speed, and to achieve memory and CPU efficiency, Kareem et al. [63] utilized the feature selection method and developed a model using the Partial Decision Tree Algorithm (PART). Initially, the authors developed various models based on REP Tree (REPT), Random Tree (RT), RF, and PART classifiers and trained them with the CICIDS2017 dataset, where RFE was used to identify the seven most important features of the dataset. For training and testing, 50:50 data was used. While a different dataset (CICDDOS2019) was used for validation. The result shows that all the classifiers achieved over 99% accuracy, where PART performed best in terms of accuracy and testing time.

Otokwala et al. [92] proposed a Common Feature Technique (CFT) based on an ensemble feature selection approach for a lightweight intrusion detection system that reduces the computation time and memory usage significantly. Here, the authors used Chi-Squared, Information Gain, and Gini-index feature selection techniques and ranked each feature of the "Water Tank" and "Gas Pipeline" datasets in order of importance. To eliminate the less important and redundant features, the authors used the returned cumulative variance values for each technique for the features. A threshold is also assumed for the cumulative variance. This produces three datasets: the Chi-Squared dataset, the Information Gain dataset, and the Gini-index dataset from the Water tank and Gas Pipeline datasets. By selecting a subset of the features common in all three datasets, a fourth dataset is generated. There were 18 features for each of the three datasets, based on the threshold, but the CFT had 14 features. The performance of CFT was compared with these three feature selection approaches with SVM, RF, and KNN, with both the Water Tank and Gas Pipeline datasets. The result shows that, even though the accuracy of CFT is a bit lower compared to other approaches, its computation time and memory usage reduce significantly with high confidence values (95% and 90%). Specifically, CFT requires 20.86 MB and 5.96 s, while Information Gain, Chi-Squared, and Gini-index require 23.79 MB, 23.71 MB, and 23.73 MB, and 6.54 s, 7.37 s, and 7.46 s, respectively.

4.3.2. Model optimization

Different optimization techniques, such as pruning, gradient compression (quantization), model selection, and hybrid algorithm optimization, can be utilized to achieve memory efficiency in ML-based network security.

Pruning and Micro-batching: Model pruning is a key technique in model compression, where the model is enhanced by removing certain

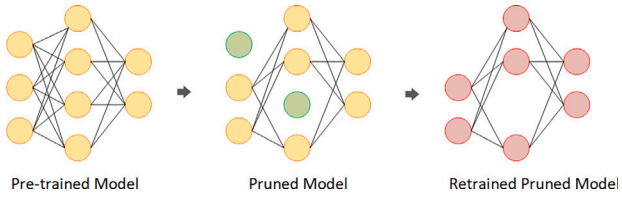


Fig. 7. Pruning algorithm application.

weights. This process can reduce the number of weights significantly, without notable degradation of the model performance. This is a simple method that can yield impressive improvements in model speed and streamline [132]. Fig. 7 shows a generic pruning algorithm, where certain nodes in the NN layer are pruned. Micro-batching is a concept that involves processing data in smaller, manageable chunks instead of handling it all at once or one piece at a time. This approach falls between batch processing, which processes data after a certain amount has been accumulated, and stream processing, which processes data as soon as it arrives in real-time or near real-time. It is an efficient way to handle large datasets by reducing latency and enhancing scalability. By dividing large datasets into smaller, parallel batches, micro-batching leads to faster and more precise processing. More about micro-batching is available [133]. These two concepts together can improve resource utilization in ML-based network security solutions.

Zakariyya et al. [80] presented Robust Effective and Resource-efficient DNN (REDNN), which is effective in attack detection and can minimize resource usage by reducing the complexity of Fully Connected Neural Network (FCNN). The approach employs simulated micro-batching, pruning, and parameter optimization to regularize the DNN model and reduce its memory and time consumption with increased accuracy. Experimental results show that this DNN model, even in FL environments, improves attack detection and resists adversarial disruptions better than benchmark baseline models and traditional ML methods commonly used in IoT security monitoring. For evaluation, the authors used eleven datasets to compare the performance with various baseline benchmark models and ML methods. The result shows that this approach can save 79.54% and 21.91% memory and time consumption compared to the benchmark, baselines in simulation, and 6.05% and 15.84% for realistic test-bed settings, with improved accuracy.

Gradient compression - Quantization: Gradient compression approaches, such as quantization, have become crucial for enhancing the deployment of neural networks on resource-constrained devices [134]. This involves reducing the precision of neural network weights and activations from higher bit-widths, like 32-bit floating point, to lower bit-widths, such as 8-bit integers. This method improves computational efficiency and memory utilization, while also boosting inference speed without significantly compromising accuracy [135]. Sabrina et al. [96] introduced a secure gradient exchange algorithm for distributed IDS in 6G networks, employing quantization to improve memory and resource usage. The algorithm integrates federated learning, blockchain, and multi-party computation for secure, decentralized learning. It uses adaptive secure aggregation to optimize computational complexity and communication overhead, ensuring efficient resource allocation and QoS in 6G networks. The FBMP-IDS architecture combines CNNs and multi-head attention for intrusion detection, achieving an average accuracy of 79.92%, a detection rate of 77.41%, and a false positive rate of 2.55% on the CICIoT2023 dataset.

Optimize with Hybrid Algorithm: Using only basic ML models for IDS can lead to unpredictability and decrease effectiveness. Adopting a hyperparameter optimizer (HPO) can improve the detection accuracy and robustness [136]. Some of the available HPOs in the literature are random search (RS), particle swarm optimization (PSO), grid search (GS), GA, hybrid bat algorithm (HBA), Tree Parzen estimator (TPE), and cuckoo search (CS). Solomon et al. [111] proposed RF with a modified HBA-based hyperparameter tuning to detect MITM attacks in the advanced metering infrastructure (AMI) of the smart grid. The authors selected RF because it outperformed other baseline algorithms in their experiment. To improve detection accuracy, various optimization algorithms were applied to RF, and their performances were compared. According to the experimental results, the proposed hybrid bat optimization (URFHBO) outperformed Bat, CS, FA, GA, GWO, PSO, TPE, GP, RS, and GS methods in terms of accuracy, memory usage, and execution time.

Model Selection: Xin-Wen et al. [78] investigated Pareto-optimal ML models to determine if there are models that are effective in terms of both accuracy and efficiency. The authors considered an IoT anomaly detection scenario, defining Pareto-optimality in terms of detection accuracy, execution time, and memory consumption. The aim was to allow individuals and organizations to select the best ML model based on their specific needs from the examined models. The authors evaluated Ridge Regression, Elastic Regression, Lasso Regression, DT, NB, RF, GBM, and DL algorithms. They found that GBM, RF, and DT are the fastest and most accurate ML models for IoT anomaly detection, while also consuming the least system memory. The accuracy, execution time, and memory usage for GBM were 100%, 19.612 s, and 103.8 MB, respectively. For RF, they were 99.4%, 4.541 s, and 81.9 MB, and for DT, they were 99.999%, 2.021 s, and 86.3 MB. Although Ridge Regression, Elastic Regression, and Lasso Regression had high accuracy, their execution time and memory usage were very high. Naive Bayes (NB) had lower execution time and memory usage, but the accuracy was slightly lower. Finally, DL had lower accuracy and higher execution time, and memory usage.

4.3.3. External facilitation

This part discusses the articles where the memory consumption reduction of ML techniques was achieved through an external facilitator.

Integration with Rules-based Implementation: ML can be implemented with rule-based security to reduce CPU usage, memory consumption, power requirements, and detection time. Kponyo et al. [66] proposed a lightweight and host-based technique for anomaly DoS detection for IoT devices through the application of ML. The technique was developed based on heuristics and NetfilterQueue and can tackle different DoS attacks. NetfilterQueue is a mechanism used in Linux for packet filtering. However, rules to match packets here are set manually and hence not scalable. The authors implemented the defence mechanism by automatically matching packets based on the output of a proposed DoS detection algorithm. The DoS technique was developed based on certain characteristics of DoS attacks observed on the end devices. Fig. 8 provides an overview of the entire detection and defence process. Here, the training was done off-device, exported, and then used on devices. For the experiment, a LAN was used as the base with four IoT devices, with one malicious IoT device. To determine the behaviour of flood attacks, the authors analysed the inter-packet arrival times and also correlated the flood attack arrival rate and their effect on the CPU and memory utilization of the end device. The authors observed that when the arrival rate increases, it significantly impacts memory and CPU. Hence, the impact on CPU and Memory was used here as heuristic parameters. The experimental results show that the method is CPU, memory, and power efficient. Its average memory usage is 23 MB, and its power consumption is 35.8 mW. Additionally, its detection time is faster than that of a conventional DoS detection method, while its average detection and mitigation times are 0.10 s and 0.22 s, respectively. Furthermore, it has 100% accuracy, which is superior to some of the works available in the literature. However, since the technique is limited to off-device training, the technique may not produce accurate results in IoT device behaviour change.

Per-Host detection: The periodic communication of a device during benign operation with a specific server can be used to develop an IDS for resource-constrained IoT devices. This approach can reduce the memory and processing time of the IDS while representing the

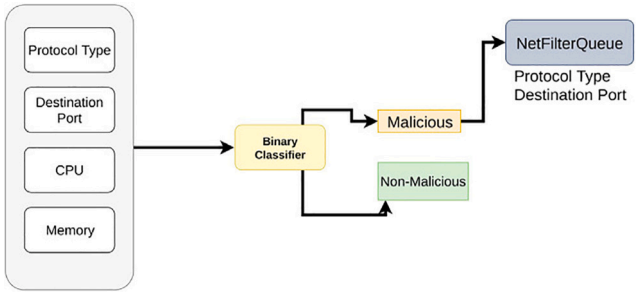


Fig. 8. ML with Rule-based denial of service (DoS) detection [66].

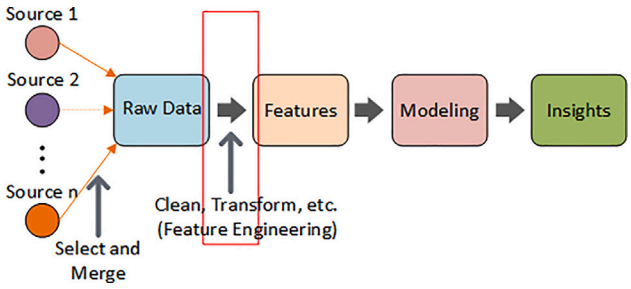


Fig. 9. Feature Engineering in ML life cycle diagram.

communication behaviour of each host using multiple entropy features. Katsura et al. [82] developed an IDS system for resource-constrained IoT devices based on the communication behaviour of the IoT devices. Here, the input data was reduced by adopting a per-host detection unit. Hence, the diversity of the flow pattern did not affect memory usage and maintained a constant value, which reduced memory consumption. Additionally, the authors used fewer features to reduce the processing time during intrusion detection. The result shows that this method can reduce memory usage by 331 MiB and processing time by 28.7 ms compared to some of the existing methods while maintaining 99.8% accuracy.

4.3.4. Key takeaways

Techniques for reducing memory consumption in ML-based network security can be categorized into data processing, model optimization, and external facilitation, with model optimization receiving the most research attention. Model optimization techniques involve model compression techniques, such as pruning that removes unnecessary neural network weights, reducing model size without sacrificing accuracy, and quantization that reduces the precision of weights and reduces memory and time consumption. Hybrid optimization algorithms can improve detection accuracy while reducing memory consumption. Model selection based on Pareto-optimality helps choose models that balance accuracy, memory, and time. Data processing techniques such as feature selection were very commonly used in memory consumption reduction. Another data processing technique micro micro-batching, was also used, with pruning for model optimization, and memory and time reduction. Finally, the external facilitator got the least focus, which involved saving memory and energy by integrating ML with rule-based implementation. Per-host detection was another facilitator that reduced memory and time by isolating traffic behaviour per device.

4.4. Time

In this subsection, we discuss various data processing, model optimization and external facilitators that reduce the time consumption of network security solutions. Table 7 provides the performance of the ML algorithms, dataset, and other important information about those network security solutions.

4.4.1. Data processing

In various solutions, time efficiency is achieved with different steps that generally fall into the preprocessing steps, for instance, feature selection, feature extraction, and traffic reduction. We included those solutions in the data processing part.

Feature Engineering: Feature engineering is a process where humans and computers collaborate to solve problems using data. It involves the modification of data based on human experience and intuitions to make the problem easier for the computer. Whereas ML is more about letting the computer solve the problem by utilizing data with minimal human help. Feature engineering involves a fair amount of trial and error, hence it is a slow process. And its success is not guaranteed [137]. Feature engineering involves various processes, such as feature creation, transformation, extraction, selection, and scaling. Fig. 9 shows where in the ML life cycle the feature engineering occurs. In various network security solutions, feature engineering was utilized to reduce training time. For instance, Aboueata et al. [85] utilized feature engineering and tuning techniques to obtain the optimal sets of features for maximum accuracy with reduced complexity, and training time while developing an IDS. The purpose was to address the data security concern in distributed multi-cloud environments. The authors found significant improvement in the accuracy of ANN and SVM algorithms by utilizing only five optimal features, where the ANN performed slightly better than the SVM. The training time was reduced as a result of dimensionality reduction in the preprocessing. Kamaldeep et al. [61] proposed a feature engineering and DDoS detection framework, demonstrating that substantial feature reduction optimizes the ML model performance with a lower amount of computation time. The framework consists of two phases. In the first phase, algorithms are developed to enhance the dataset and focus on advanced feature engineering to statistically analyse the data, considering probability distributions and feature correlations. In the second phase, the authors introduced an ML model and conducted a complexity analysis of the feature-engineered dataset, using five ML techniques: SVM, LR, DT, MLP, and RF. In the experiments, RF performed the best in detecting all types of attacks. LR required the least computation time at 145 ms, but did not perform well in terms of detection. RF required 176 ms, making it the second fastest, while MLP required the highest computation time at 4541 ms.

- **Feature Selection:** Feature selection aids in reducing the dimensionality of the dataset by eliminating less relevant features. As a result, the ML model needs to process less data, leading to a reduction in processing time and the use of other important resources. This technique is widely used in various ML-based network security solutions to reduce latency and training time. Many researchers are also attempting to develop novel feature selection algorithms to reduce computational complexity and latency. Linhore et al. [88] utilized a hybrid feature selection method in their proposed Network Intrusion Detection System (NIDS) to minimize the complexity, training time, and prediction latency of the NIDS, without compromising prediction efficiency in healthcare security. The hybrid feature selection method was developed by integrating SVM and the Least Squares methods. The performance was compared with SVM and KNN models, and the experimental result shows that the hybrid method outperforms SVM and KNN in terms of training time, AUC, and precision.
  Dubey et al. [138] proposed two optimal feature selection approaches for ML-based IDS. These approaches aim to reduce the dimension of the dataset, thereby reducing the training time. The methods were developed based on Mutual Information (MI) and Kendall's Correlation Coefficients. The first method, Dense-FR, ranks features using Mutual Information (MI) and Kendall's

Table 7
ML-based network security techniques that improve efficiency with respect to time or latency.

Ref	ML Algorithms*	Datasets	Time efficiency achieved by	Validation techniques	Acc.	Pr.	Re.	FS
[85]	ANN	UNSW-NB-15		Holdout	92	93	92	92
[90]	NB	MQTT	Feature Engineering	Holdout	64.05	59.99	80+	74+
		UNSW-NB15			77.35	86	90+	73+
		IoTID20			25.18	100	20+	98+
[75]	RF (low-end)	Custom		Cross-validation	92.98	94.74	–	96.29
	DT (high-end)				92.23	96.45		95.76
[67]	DT	Custom	Feature selection, hyper-parameters fine-tuning	Holdout	99	99	97.5	–
	Rule-based		Feature selection, heuristic rules Implementation		100	100	100	
[138]	NB	KDD 99	Feature selection, Dense FR	Cross-validation	94.59	94.03	98.2	96.04
			Feature selection, Sparse FR		94.96	94.76	98.2	96.41
[87]	Facebook Prophet	CSECICIDS2018	Feature selection	Cross-validation	98.30	–	–	98.90
[88]	Hybrid	–		Cross-validation	100	95	100	100
[57]	ELM	NSL-KDD		Cross-validation	91.7	–	–	–
[91]	DT	UNSW-NB15	Feature selection	–	–	80.18	76.62	78.36
	MLP		Feature extraction			79.90	71.36	75.39
[82]	RF	Bot-IoT	Feature extraction	Cross-validations	99.8	–	–	–
[61]	RF	IoT-CIDDS (Cooja)		Cross-validation	98+	98+	98+	97+
[107]	RL	–	Network traffic reduction	Cross-validation	98.30	–	–	98.90
[89]	RF	KDD99	Model selection	Holdout	99.95	–	–	–
		NSL-KDD			99.87			
		UNSW-NB15			97.43			
		CIC-IDS-2017			99.81			
[77]	FL(DRL)	–	Decentralized learning	Stratified sampling	–	–	–	–
[106]	SVM	Custom	Packet batching	–	86	–	–	86

ML Algorithms*: Best Performing ML Algorithms, Accuracy: Acc., Precision: Pr., Recall: Re, F1-score: FS.

Correlation Coefficients, then selects common features from the top-ranked subsets. The second method, Sparse-FR, uses only the top 60% of features. The performance was evaluated with KDD, IG-C-3, and the two proposed approaches, where KDD had 41, Dense-FR had 20, and Sparse-FR had only seven attributes. Three ML algorithms were used for the evaluation: they are MLP, NB, and LR in terms of accuracy, precision, recall, f1-score, Jaccard score, and hamming loss. The results show that the approaches work well with NB and the One-vs-Rest classifier using LR. However, the result with MLP was inferior comparatively.

Al-Ghuwairi et al. [87] employed a feature selection method in the IDS for cloud computing with time series data that significantly reduces the training time, complexity, and accuracy. The strategy involves a feature selection method and a prediction model based on Facebook Prophet. The feature selection method merges time series analysis with anomaly detection, stationarity, and causality tests to confront the problem of false associations between anomalies and attacks. The approach was evaluated with the CSE-CIC-IDS2018 dataset, while data cleaning was carried out on the dataset using the IDS-Dataset-Cleaning tool at the preprocessing step. The proposed feature selection method involves three steps: in the first step, the stationary columns were selected from the dataset, and other columns were also converted to stationary by applying the KPA test. In the second step, only the columns that have Granger causality in their label column were selected. In the third step, the column where the anomalies showed Granger causality in their label column was selected. However, the Facebook prophet-based model was constructed using lag predictors to project the values of the label column, which serve as indicators of bot attacks. The experiment result shows that, although the memory usage remains the same, this approach reduces the training, testing, and cross-validation time to 85%, 15%, and 97%, respectively.

Yusof et al. [57] proposed an adaptive feature selection algorithm for DDoS detection that improves training time and accuracy and can reduce the computational complexity in data analysis while the attack occurs. The algorithm was developed by combining Consistency Subset Evaluation (CSE) and DDoS Characteristics Features (DCF) algorithms. For the experiment, the authors used the NSL-KDD dataset, and the performance was compared with Chi-squared, Gain Ratio, Information Gain, and CFS feature selection algorithms. The Extreme Learning Machine (ELM) algorithm [139] was used for model development, which is a bit different from the traditional feed-forward neural network. In ELM, the parameters in the hidden layer are selected randomly and are not tuned. The result shows that the proposed algorithm can achieve 91.7% accuracy with the shortest training time, which is 1.469 s. The Chi-squared method achieved the highest accuracy with 96.2%, but it needs 2.036 s for training. Additionally, the Gain Ratio achieved 92.1% accuracy, but it took 81.797 s for training, which is the highest training time. Finally, CFS and Information Gain had comparatively lower accuracy and higher training time.

- **Feature Extraction:** Feature extraction is a crucial technique in ML and involves converting unprocessed, unorganized data into a collection of the most relevant attributes that an algorithm can leverage for classification or prediction. Especially, it creates new features that encapsulate the most relevant and crucial information from a dataset [140]. Eq. (2) shows the visual representation

of the feature extraction. Here, $X_1, X_2, \ldots, X_N$ represent an input vector of N features. Here, f represents the specific method or algorithm used to extract the new features from the input data. $Y_1, Y_2, \ldots, Y_M$ represent the output vector of M extracted features. The number of extracted features M could be less than, equal to, or greater than the number of original features, N, depending on the feature extraction method used.

$$\begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_N \end{bmatrix} \xrightarrow{\text{feature extraction}} \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_M \end{bmatrix} = f\left(\begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_N \end{bmatrix}\right) \tag{2}$$

Sen et al. [113] utilized Principal Component Analysis (PCA) or Linear Discriminant Analysis (LDA) feature extraction techniques in the first stage of their two-staged IDS for smart grid false data injection. Here, PCA and LDA are utilized to reduce the data dimensionality. In the second stage, the classification operation is performed using the SVM. According to the evaluation result, LDA-based SVM had better accuracy and reduced simulation time compared to only SVM and PCA-based SVM setups.

Ngo et al. [91] conducted a performance comparison between the feature extractions and feature selections. The authors used the UNSW-NB15 dataset with binary and multi-class classification scenarios. The comparison result shows that the feature selection performs well and requires less training time when the number of reduced features (K = 8 or 16) is large. Meanwhile, the feature extraction method performs well when the number is small (K = 4 or less). Additionally, both binary and multi-class feature extraction are less sensitive to the change in the number of reduced features and can detect a wide range of attacks. In addition, both can detect more attacks when more features are selected or extracted. DT also performs best with the feature selection method, and MLP is the best classifier for the feature extraction, while the comparison was done between MLP, DT, NB, RF, and KNN. Furthermore, the authors provided some guidelines for choosing an IDS type based on different scenarios.

Traffic Reduction: Traffic reduction is the process of data management and optimization applied to network traffic data. The network traffic is divided into time windows and analysed in each window separately. Since the volume of the data to be processed at any given time is reduced, the data analysis process becomes more efficient, reducing the amount of time required for processing. Alauthman et al. [107] suggested a botnet detection method that uses RL and includes a traffic reduction technique. This method improves data analysis efficiency by reducing network traffic data, which significantly reduces the training time and improves the learning rate of the newly extracted features in the online system. The process involves capturing and reducing network traffic, extracting features from the reduced traffic for botnet detection, and using RL for malicious activity and botnet detection. The system uses three NNs for botnet detection, with the best-performing NN used in the detection phase. The system demonstrated significant efficiency, with a 40%–70% reduction in traffic and high accuracy rates.

4.4.2. Model optimization

By decentralizing learning and selecting appropriate machine learning models based on time efficiency, the time consumption of ML-based network security solutions can be reduced.

Decentralized Learning: Decentralized learning can improve latency and throughput of network security. Wang et al. [77] presented a two-phase anomaly detection framework, named federated deep reinforcement learning empowered anomaly detection (FLAD), to address the privacy leakage challenge in IIoT. By utilizing decentralized learning, the framework can detect anomalies with improved throughput and latency. The framework consists of four entities, they are the Global Anomaly Detection Center (GADC), the Regional Anomaly Detection Center (RADC), the Local Anomaly Detection Center (LADC), and the Users. Here, a global model is developed using the FL model, while the local models are trained with deep reinforcement learning (DRL). FLAD detects abnormal users who may give away the victim's identity, position, or other sensitive information by their activities. Since each user belongs to a specific region, the anomaly detection is done by the corresponding RADC for the internal users. Additionally, to improve anomaly detection accuracy, the model performs anomaly detection on the anomaly detection centres (ADC) first. Also, an appeal mechanism was introduced for the misjudged users to reclaim the non-anomaly. The authors defined three cases based on different user anomaly rates and RADC anomaly rates, set at 5%, 10%, and 15% for both. The results show that although latency increases with the number of users, the average latency for each case is 13.5 s, 11 s, and 9 s.

Model Selection:

This segment explores studies where certain ML algorithms were determined to be time-efficient in various comparative evaluations. Leon et al. [89] compared five supervised algorithms, which are ANN, SVM, RF, LDA, and KNN, and three unsupervised algorithms, Mean-shift, K-Means, and DBSCAN. The datasets used were KDD99, UNSW-NB15, NSL-KDD, and CIC-IDS-2017, where the datasets have three different kinds of variables: binary, real, and categorical. The experiment result shows that RF has the highest accuracy among all the models, and the reason behind this better approximation is the use of more than one model. For the CIC-IDS-2017 dataset, KNN performed the best, but it also used a longer execution time. Additionally, although LDA, K-means, and Mean shift algorithms take a short time compared to other algorithms, the accuracy was comparatively lower.

Jien et al. [90] evaluated various data sets for IoT IDS applications and compared the performance of various ML algorithms based on those data sets. The work aimed to provide recommendations for the best ML techniques for IDS based on F1-score, accuracy, and efficiency. Seven ML algorithms were used here: they are NB, KNN, DT, SVM, RF, CNN, and ANN, along with three data sets: IoTID20, UNSW-NB15, and MQTT. All of the ML models were based on supervised learning, and the datasets were processed by transforming the data so that the model could use them and remove irrelevant features. The results indicated that DT was the most suitable ML algorithm in terms of accuracy and F1 score, scoring the highest average points. Additionally, it was the second most efficient in terms of training and testing time, which were 0.53, 1.04, and 16.31 s for the MQTT, UNSW-NB15, and IoTID20 datasets, respectively, with accuracies of 95.67%, 89.49%, and 99.92%. Although NB was the fastest in training and testing time, its accuracy and F1 score were low, taking 0.29, 0.19, and 1.56 s, respectively, with accuracies of 64.05%, 77.35%, and 25.18%. Meanwhile, SVM had the highest training time among all the algorithms, taking 922.06, 681.26, and 98.85 s, respectively. The accuracies of SVM for these datasets were 76.3%, 66.41%, and 95.99%, respectively.

4.4.3. External facilitation

By integrating external facilitators such as rules-based implementation and SPARK with ML, the time efficiency of network security can be improved.

Integration with rule-based implementation:

A heuristic, or a heuristic function, is a method used in search algorithms to prioritize alternatives at each decision point. It uses the information currently available to determine the next step to take. In some cases, it might provide an estimated solution rather than an exact one [141]. This approach assists the algorithm in identifying a suitable selection, eliminating the need for a comprehensive search of all potential solutions. As a result, it can expedite the finding of an approximate solution. It is a shortcut technique and, hence, may result in sub-optimal system performance. Rahbani et al. [67] attempted to develop an optimal volumetric DDoS detection system for IoT by utilizing heuristic-assisted rules, with the help of ML to reduce the detection

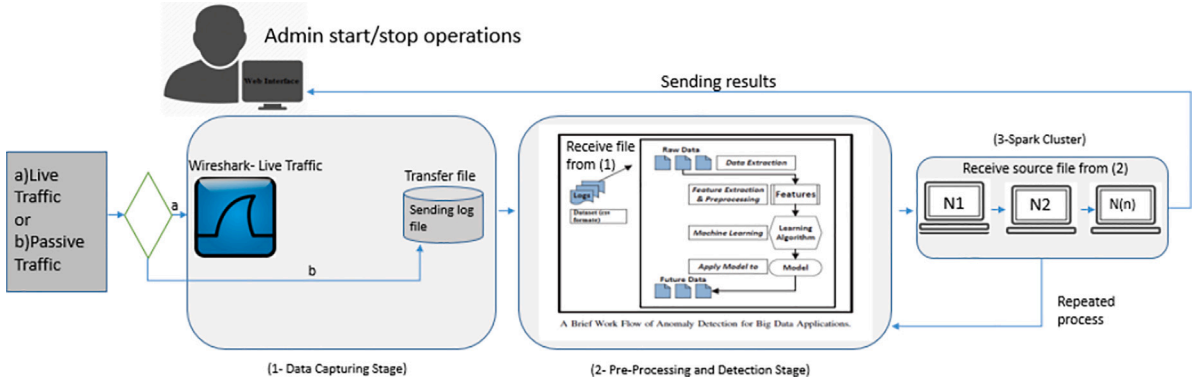


Fig. 10. Spark-based anomaly detection framework (SAD-F) [75].

time. Two simple heuristic rules were created for DDoS detection: improving performance and reducing classification time with the help of ML feature selection and human assistance. Based on the experiment, the heuristic method performed better than DT in all the performance matrices. Additionally, the classification time for the heuristic method is 56.12% less than the DT Algorithm.

Implementation of SPARK with ML:

Apache Spark [142] is a cluster computing platform, developed with the goal of providing a near real-time response. It can handle various types of computation, such as interactive queries and stream processing. The platform's speed is crucial for processing large datasets. Ahmed et al. [75] presented SAD-F, an intelligence DDoS identification framework for enterprise networks, which utilizes the Apache Spark framework for security analysis. This framework was able to analyse potential DDoS attacks without any delay in live and passive traffic. It captures live traffic and extracts relevant information summary by pre-processing this traffic and utilizing ML-Spark algorithms for running detection algorithms for DDoS. The framework supports five supervised ML algorithms: SVM, NB, KNN, DT, and RF. Fig. 10 shows the SAD-F framework, where the Spark Cluster is the third phase in the framework. It has five working nodes, where One of the nodes works as a master, and the others as slaves. Its processing runs on memory instead of disk, which is one of the essential features that allow it to achieve its computation speed. The authors found that the framework can detect DDoS attacks with minimal delay when the performance was checked with live and passive traffic. Additionally, this SAD-F can solve memory inefficiency, scalability, and complex processing efficiently and with low latency by utilizing parallel data processing. Finally, before implementing the SAD-F, it was tested on a Local Machine, and the result showed that it can train and test overall files with 92% accuracy, approximately within 3.5 h in the worst-case scenario with KNN; whereas it took 1.8 h for the best-case scenario with 72% accuracy for the NB classifier. These tests were done with a file size of 150 MB, which is about 1300K packets of data. Meanwhile, the SAD-F in the two different testbed setups was much faster. The first setup was a low-end configuration, where RF processed 100K and 3400K packets in 5.0001 s and 1075.6666 s, respectively, whereas in the high-end configuration, DT processed 100K and 3400K packets in 4.3333 s and 191.0001 s, respectively, as the best case. The authors also noticed that capturing data uses more memory than processing. However, increasing the size of the cluster can make better use of the CPU, which makes the whole system efficient.

4.4.4. Key takeaways

The techniques used to reduce time consumption in ML-based network security can be categorized into data processing, model optimization, and external facilitation. Among these, data processing received the most research attention, followed by model optimization and then external facilitation. Data processing techniques included feature engineering, such as feature selection and feature extraction, which help reduce data dimensionality and improve training and classification speed. Another important technique was traffic reduction, which decreases the volume of data to be processed, enhances learning rates, and shortens processing time. Model optimization involved decentralized learning, which improves latency and throughput by distributing learning across different layers—local, regional, and global. It also included model selection, where comparative studies identified time-efficient models that maintain high detection accuracy. External facilitation included rule-based integration, where heuristic-assisted ML significantly reduced detection time compared to traditional ML approaches. Additionally, the integration of Apache Spark enabled near real-time detection through parallel processing and in-memory computation, resulting in significantly lower latency.

4.5. Bandwidth and or spectrum

A few researchers have developed network security solutions that achieve effective security measures with reduced bandwidth utilization. In this subsection, we discuss various bandwidth-efficient solutions. Table 8 presents their performance, datasets, and other important information.

4.5.1. Model optimization

An autoencoder (AE) is a specific type of ANN that takes high-dimensional input data and compresses it to a meaningful and compact format, and later uses this compressed data to recreate the original input data. AEs generate their own labels and are known as self-supervised learning algorithms. The AEs consist of three components: an encoder, code (latent space), and a decoder [143] as shown in Fig. 11. Huong et al. [72] proposed an anomaly detection system, which is a hybrid design of Federated learning, autoencoder, transformer, and Fourier (FATRAF) mixing sub-layer that can reduce the bandwidth consumption significantly. It was developed to detect the irregularities in the time series data in the industry and minimize the threat to smart manufacturing. It is a lightweight local learning model that requires less training time and other computational resources and can be implemented in distributed edge devices in an IoT-based industrial control system. The FATRAF leverages the FL and sends only the learnable parameters of the Transformer-Fourier block to the cloud server in each round instead of raw data. Moreover, the autoencoder training phase is entirely implemented within the local site, and hence, clients do not need to exchange any information during this phase. Finally, during the Transformer-Fourier training phase, the model's weight matrices are transferred to the cloud for aggregation when the local training is done. By ensuring efficient communication, this method reduces the communication rounds effectively. Hence, FATRAF achieves reduced

Table 8
ML-based network security techniques that minimize the consumption of bandwidth.

Ref	ML Algorithms*	Datasets	Bandwidth efficiency achieved by	Validation techniques	Acc.	Pr.	Re.	FS
[118]	RF	CIC DoS	Implementation in specialized networking architecture (ML with SDN)	Cross-validation	96+	–	–	92+
[72]	FL	Gas Pipeline	FL, VAE	Cross-validation	–	96.83	100	98.39
		SWaT				93.89	97.75	95.78
		HAI				89.39	100	94.40
		Power Demand				92.85	94.42	93.63
		ECGs				98.56	100	99.27
		Respiration				95.78	97.30	96.54
		Gesture				93.31	99.06	96.10
		Space shuttle				97.38	100	98.67
		NYC taxi				94.19	100	97.01
[73]	FL (Hybrid)	Space shuttle		–	–	100	100	100
		Respiration				93.13	55.30	69.39
		Gesture				52.74	100	69.10
		NYC taxi				96.06	100	97.99
		ECG				100	100	100
		Power demand				73.55	91.00	81.35
		SCADA				96.09	99.82	97.92
[94]	DT and DL	CICIDS-2017	Task assignment and capacity allocation	Cross-validation	–	–	–	–
[95]	DQL	NSL-KDD	Resource allocation	–	94	–	–	80.84

ML Algorithms*: Best Performing ML Algorithms, Accuracy: Acc., Precision: Pr., Recall: Re, F1-score: FS.

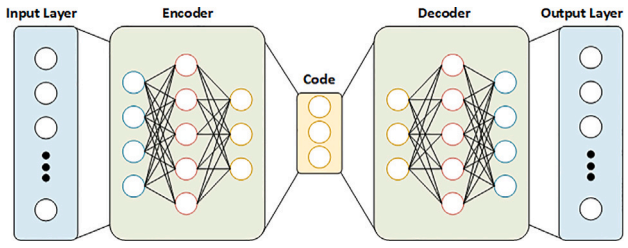


Fig. 11. A conceptual representation of an auto encoder.

bandwidth consumption. The bandwidth usage of FATRAF is generally low. For example, with the smallest hidden dimensionality (dhidden = 1), the average uplink and downlink usage are around 0.32–0.73 kiB/s and 0.26–0.62 kiB/s, respectively. Finally, it consumes very little CPU, memory, and learning time; hence, it can be deployed in resource-constrained edge devices.

In another work, Huong et al. [73] presented an FL-based IIoT decentralized architecture to prevent cyber attacks in IIoT-based smart manufacturing that can reduce bandwidth usage. The presented FL-based IIoT decentralized architecture can detect cyberattacks in the industrial control system by detecting malicious time series data in the industrial system. Here, decentralized anomaly detection is performed at the edge, where a hybrid device of VAE and LSTM is deployed. This facilitates it to cope with anomaly detection for time series data. For achieving high accuracy, it utilizes KQE. Moreover, the VAE module in this hybrid model enables it to compute fast enough with higher accuracy and makes it suitable to be deployed at the edge. In the end, FL is used to send each edge device's trained model to the cloud for a global update. This approach helps to decrease the bandwidth usage between the edge and the cloud. Moreover, it allows the detection system to remain at the edge for quicker response to attacks while ensuring that each edge device's trained model, which monitors a specific area, is globally updated. The results from the experiment showed that this model is efficient and feasible to deploy in smart manufacturing and can save 35% of bandwidth. Since this architecture uses a maximum of 85% of CPU and 37% of memory, it is realizable in IIoT-based smart manufacturing.

4.5.2. External facilitation

Various external facilitators, such as the utilization of specialized network architectures, effective task and capacity assignment, and proper resource allocation, can aid in achieving bandwidth efficiency for machine learning-based network security solutions.

Specialized Networking Architecture:

Even though an HTTP flood attack is an application-layer DDoS attack, it makes the web servers unavailable and consumes significant network resources. SDN enables the network providers to be able to detect and mitigate these types of attacks. However, the combination of SDN and ML can be more effective and resource-efficient. Mohammadi et al. [118] proposed HTTPScout by combining the benefits of both SDN and ML, which can detect and mitigate HTTP Flood attacks while reducing bandwidth consumption significantly. Since the SDN controller has a global view of the network, it can gather all the statistics about the network flow. On the other hand, by applying ML algorithms, the normal and abnormal traffic can be separated. Hence, in HTTPScout, a supervised ML algorithm trained with normal and abnormal traffic is used along with SDN. It monitors all the traffic flow from the network and extracts the important features from each flow. These features are then fed into the ML algorithm. If an HTTP flow is identified as an attack based on the algorithm's training data, the source of the attack is pinpointed and blocked by implementing rules on the network's edge switch. The authors used RF, KNN, DT, and NB algorithms to compare the performance of HTTPscout, where RF showed superior performance in terms of accuracy, F1-score, and average bandwidth consumption. Additionally, HTTPScout improves 64% of bandwidth consumption and 80% of the forwarding rules compared to the traditional SDN. Finally, it also reduces the memory waste in the OpenFlow switches and can be implemented in a modular fashion.

Task Assignment and Capacity Allocation: A scalable multi-tier IDS architecture consists of bottom, middle, and top tiers. By distributing computation across these tiers, this architecture can significantly reduce bandwidth and latency by bringing the computation closer to the users. Lai et al. [94] presented an IDS that can detect zero-day attacks with ten different task assignment scenarios using queuing theory. The task includes pre-processing, binary attack detection, and multi-class attack detection, which were assigned to fog, edge, and cloud. The authors assessed different tiered architectures where the task was assigned to either one, two, or all three of the fog, edge, and cloud, while the task performed in either one of them is called

one-tier, for either two is two-tiered, and divided among all three of them is three-tiered architecture. The queuing algorithm calculates the total delay, and the simulated annealing approach to allocate the optimum capacity for each tier. A DT algorithm was used here for binary detection, and a DNN for multi-classification. Here, the binary classification only classifies the traffic as normal or malicious, where only malicious traffic is transferred to the multi-class to determine the attack class, while a flow-based approach is employed for traffic analysis. The authors found that the combination of fog and edge can reduce bandwidth consumption significantly because here, the traffic is processed near the source without requiring it to be sent to a centralized location. Moreover, it improves user privacy and also a 16% reduction in delay. In the combination where the pre-processing and the binary classification were done in fog, the multi-class detection to the edge required 1.05 Gbps, while the edge–cloud combination required 5.29 Gbps. The pre-processing and binary detection occurred in the edge and the multi-class at the cloud. However, the three-tier, which combines cloud, fog, and edge, is not suitable, since it has higher latency and requires a lot of federation. The IDS has lower latency if all the tasks are performed in the edge or in the edge–cloud combination. Finally, the result shows that the algorithm assigns 85% of the total capacity to the lower-tier for pre-processing to reduce delays.

Resource Allocation: By allocating bandwidth only to trustworthy devices, proper utilization of bandwidth resources can be ensured. Hajar et al. [95] introduced an IDS for IoT security in 5G and beyond networks, where the bandwidth consumption is reduced by optimal resource allocation. The authors employed a Deep Q-Learning (DQL) based approach, where the algorithm predicts anomalous actions by observing the behaviour of the connected devices. A reputation mechanism-based bandwidth allocation technique is also introduced, which evaluates the trustworthiness of the participating IoT devices by analysing their behaviour and interaction. The IDS was developed based on a four-layer DRL architecture, with a ReLu action function. The input layer on the top captures environment variables, two intermediate hidden layers support the training, and the final layer represents the Q-values for attack classification. A 500 square metre area centred by a BS was considered with m number of uniformly distributed devices for the simulation, which resembles both cellular and mobile networks. The simulation result shows that the approach can achieve an AUC of 0.88 after 50 epochs of training, and its loss values decrease consistently with each epoch and become nearly zero during 50 epochs of training. Hence, this self-learning mechanism can be adopted and improved over time in anomaly detection. The author compared this approach with SVM, RF, Self Organization Map, CNN, CNN-BiLSTM, and Bidirectional LSTM, while it performed superiorly over the other approaches.

4.5.3. Key takeaways

Among all the KPIs, bandwidth and/or spectrum received the least attention, as shown in Fig. 2. The approaches to improving bandwidth efficiency can be broadly classified into two categories: model optimization and external facilitation techniques, with the latter receiving comparatively more focus. External facilitation methods—such as specialized networking architectures like SDN—can significantly reduce bandwidth usage, achieving up to a 64% reduction in DDoS attack detection scenarios. Additionally, distributing tasks like preprocessing, binary detection, and multi-class detection across fog, edge, and cloud layers can lead to substantial bandwidth savings. The fog-edge combination is particularly effective, as it enables data to be processed closer to the source, minimizing the need to transmit traffic to centralized cloud servers. Moreover, bandwidth efficiency can also be enhanced by allocating available bandwidth exclusively to trusted devices, ensuring optimal resource utilization. In terms of model optimization, techniques such as FL and VAE were employed. VAE compresses high-dimensional input data into compact latent representations, while FL reduces communication overhead by eliminating the need for centralized data sharing. Furthermore, the Transformer-Fourier sublayer minimizes the volume of data exchanged during training rounds. Although data processing was not categorized separately, several of the reviewed approaches, both under external facilitation and model optimization, incorporated data processing techniques to enhance data quality and further support bandwidth efficiency.

4.6. Human resource

This subsection focuses on ML-based network security techniques that reduce human involvement in preparing and managing ML-based network security solutions. Different automated processes can significantly reduce human involvement, as shown in Table 9 and discussed below.

4.6.1. Data processing

Automated data processing can help minimize human involvement in ML-based network security. Below, the most common approaches in this regard are discussed.

Active Learning:

Active learning is a framework designed to minimize the cost associated with annotation during training tasks. It aims to enhance a model's performance as much as possible while requiring the annotation of the fewest number of samples. Specifically, it aims to identify the most valuable samples from an unlabelled dataset and have an oracle (such as a human annotator) label them, thereby minimizing labelling costs while maintaining performance [145]. Fig. 12 shows an active learning paradigm. There are three main components in active learning: query strategy, annotator, and ML model. The query strategy selects unlabelled data based on a specific policy. This data is then labelled by a human or machine annotator and added to the training set. The model is subsequently updated, and this cycle continues until new data is no longer available or a stopping criterion is met. Stopping criteria can vary such as achieving a target accuracy, reaching a time limit, or hitting a maximum number of queries, all of which can influence the effectiveness of active learning [146]. Hidetoshi. et al. [104] integrated an active learning paradigm in their signature-based intrusion detection and prevention systems (IDPSs), which reduces the annotation cost for the experts. In general, for optimizing an IDPS, network security experts classify signatures by low, medium, or high importance. Even though ML can automate this process, it involves various challenges, including annotation cost, classification error, and accuracy degradation due to domain shift. With the integration of active learning, this IDPS can collaborate with experts in periodically classifying the received signatures. Here, the signatures are sorted using uncertainty sampling, some are sent to experts for review, while the rest are classified automatically. Mergendahl et al. [59] utilized active learning to deal with domain shift in IoT DDoS detection. The authors developed Rapid, which can detect real-time DDoS attacks in IoT environment while maintaining the accuracy demand under the domain shift with very little or no maintenance from the network operators. Here, the active learning integrates with the attack mitigation method and adapts to the new network domains automatically without minimal human intervention.

AutoML:

AutoML automatically performs various data analytics and machine learning processes, such as pre-processing, feature engineering, model selection, hyperparameter optimization (HPO), and model updating [147]. Where automated data processing and feature engineering are implied to improve network data quality, model selection, and HPO for model optimization, and automated model updates for dealing with drift issues and model performance [116]. Yang. et al. [116] utilized AutoML in their survey to address the challenges of ZTNs in reducing the need for human expertise in developing ML models. The experiment used two datasets, CICIDS2017 and 5G-NIDD with

Table 9
ML-based network security techniques that minimize the involvement of human resource.

Ref	ML Algorithms*	Datasets	Human-resource efficiency achieved by	Validation techniques	Acc.	Pr.	Re.	FS
[59]	LSTM	Smart Home 1	Active learning	Cross-validation	–	97	89.7	92.9
[104]	Custom	Custom		Holdout	≈90	–	–	–
[81]	Custom	Real-world IoT-23	Unsupervised learning	–	94.2	–	–	95.6
[76]	GA + Fuzzy logic	–		Evaluation metrics	96.53	95.23	76.50	84.84
[93]	DRL	NSL KDD	Context Adaptation	–	81.07	–	–	–
		UNSW-NB15			85.09			
		AWID			96.02			
[115]	RL	Custom		Cross-validation	98.63	98.21	99.07	98.64
[98]	RF	Custom	AutoML	–	95.50	95.98	97.87	–
[102]	DT	KDDcup99		Cross-validation	99.7	–	–	–
[116]	LightGBM (Offline)	CICIDS2017		Cross-validation	99.77	99.47	99.38	99.42
		5G-NIDD			99.75	99.87	99.79	99.83
[144]	Custom	Power dataset 1		Cross-validation	93.75	92.61	92.15	92.38
		Power dataset 3			94.38	95.28	92.54	93.84
		Power dataset 3			87.12	86.34	85.49	85.84
[74]	LGBM	CICDarknet2020		Cross-validation	98.96	96.70	98.96	98.96
[97]	GPR	Custom		Cross-validation	–	–	–	–
[108]	AutoML	Network Device		Cross-validation	95.4	–	–	95.5
		Dataset			99.9			99.9
		CICIDS 2017			99.9			99.7
		DTLS Handshakes			92.4			93.2
		netML IoT			81.9			79.5
		netML Non-VPN			99.9			99.9
		netML CICIDS 2017			96.8			90.4
		Cross Platform			98.4			98.6
		Streaming Video Providers						
[74]	FL, GRU	Custom	Device-type-specific anomaly detection	Cross-validation	95.43	–	–	–
[117]	K-Means	Custom	Auto parameter selection	–	–	–	–	–

Algorithms*: Best Performing ML Algorithms, Accuracy: Acc., Precision: Pr., Recall: Re, F1-score: FS.

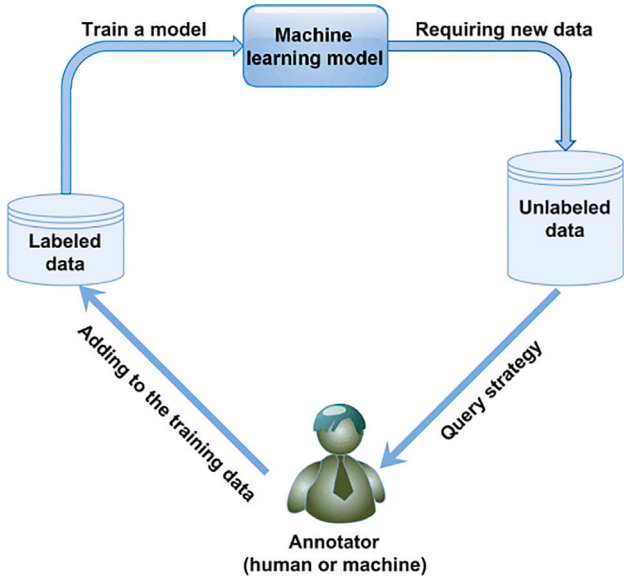


Fig. 12. Active learning [146].

various ML models. The result shows that the AutoML pipeline can generate a superior IDS with 0.134% and 0.134% improved F1-score, and 50% and 12.5% reduction in learning time compared to the best-performing algorithm without AutoML during offline training, while automating the data processing and feature engineering. Additionally, it shows greater adaptability to model drift, with its accuracy dipping and recovering at drift points. This suggests that the AutoML-enhanced model can sustain high detection accuracy despite the data volatility typical of cybersecurity threats. Jordan et al. [108] introduced a unified packet representation tool called 'nPrint.' When integrated with AutoML, nPrint can eliminate the need for manual feature engineering and model tuning for a wide range of network traffic analysis tasks. nPrint encodes each packet into a normalized, binary format that preserves the essential details of the packet. This enables the automatic identification of important features from packet sets for various classification tasks, removing the need for manual feature extraction from complex network protocols. This automation accelerates the development and deployment of machine learning in networking, making it easier to implement in practical scenarios.

4.6.2. Model optimization

Automating model optimization via AutoML, automated parameter selection, unsupervised learning, and automated context adaption minimizes the need for human interaction and, consequently, human errors.

AutoML: AutoML can save hours of ML professional's time by automating various steps, which ensures efficiency, scalability, and human mistake-free outcomes. Alexandros. et al. [109] introduced a network traffic analyser that focuses on the timely detection and effective treatment of cyberattacks to protect personal and sensitive data from unauthorized disclosure. Here, the authors utilized AutoML to automatically select the top-performing ML model among competitors for identifying threats. This aids in quicker and more precise detection of security threats, which can be suitable for a fast-changing cybersecurity environment. ML model Danish. et al. [144] utilized AutoML and sophisticated feature engineering to reduce the need for domain expertise in developing Industrial Control Systems (ICS) security. This approach, named ICS-defender, reduces the dependency on domain-specific knowledge by automating model selection, training, and optimization. The system is scalable and can detect various attacks and unknown OT-IT security threats, and allows non-experts to deploy

and optimize security solutions efficiently. Singh. et al. [97] automated the model selection and hyperparameter tuning in intrusion detection in WSN. Where the AutoML-based approach accurately estimates the number of k-nearest neighbours needed for rapid intrusion detection and prevention in WSN in a rectangular region of interest (RoI), considering a Gaussian distribution of node deployment. The authors found that synthetic predictors derived from Monte Carlo simulations effectively corresponded with the k-barriers. Additionally, AutoML selected GPR algorithm as the best machine learning model to map the required k-barriers accurately, which can perform exceptionally well on the unseen dataset, and can outperform other benchmark ML models in terms of accuracy.

Automated Parameter Selection:

Manual selection of parameters in ML may require a significant amount of time and effort, and may even achieve poor detection accuracy. By automating the parameter selection process, both manual labour and human error can be reduced. Chuanpu. et al. [117] presented Whisper, which is a malicious traffic detection system that utilizes frequency domain analysis. Where the author developed an automated vector selection to reduce the manual work in parameter selection. This process can ensure high detection accuracy while avoiding manual parameter setting. Whisper can perform real-time detection in a high-throughput network. The authors showed that, automatically parameter selection achieves 9.99% better AUC, and 99.55% better equal error rates (EER) than the manual parameter selection method.

Unsupervised Learning: Unsupervised learning does not require any labelled data, hence it can reduce a significant level of human labour. However, the unsupervised learning training dataset should consist of benign traffic only, which may be subject to human intervention [148]. Vincenzo. et al. [81] presented an unsupervised learning based multi-layered system to deal with concept drift in IoT anomaly detection. A general challenge in ML-based threat detection systems is that a change in the statistical distribution of data can cause a significant drop in performance. This phenomenon is known as concept drift. When this happens, traditional static systems need manual retraining by humans. In contrast, this multi-layered system is designed to handle concept drift by automatically retraining itself only when needed. Hamamoto et al. [76] combined GA and Fuzzy logic for network anomaly detection, which autonomously uses collected network data, does not require any data labelling, and produces a satisfactory result. Here, GA generates a digital signature using flow analysis, and the extracted data is used for the prediction of network behaviour for a given period. It extracts only six attributes from the network flow. The Fuzzy logic determines whether the instance is malicious or not.

Context Adaptation: Network environments today are complex and constantly evolving because of the new types of attacks and shifting attack patterns. To remain effective, network security solutions need regular updates through retraining with both old and new datasets. However, this is often impractical because of high computational cost and resource demand. These models also require frequent human intervention for retraining and transformation, which can introduce vulnerabilities and degrade performance [93]. If ML models can adapt to different contexts automatically, human resources can be reduced significantly. Micale. et al. [98] introduce CAHOOT, a context-aware vehicular intrusion detection and prevention system which can detect intrusion with both autonomous and human driving modes. Here, the context information is collected at run-time from the vehicle's engine and sensors. This information is used to determine the environmental data such as traffic conditions, and also the driver's habits. The system can determine various intrusion attacks like DoS, spoofing, and replay attacks, as well as any form of anomalous behaviour from the driver, like accelerating in front of an obstacle. Kamalakanta et al. [93] presented a context-adaptive IDS, which leverages Deep Reinforcement Learning (DRL) agents distributed across the network to adapt to various attack patterns. The system balances accuracy and false positive rate (FPR) by integrating ensemble techniques with deep Q-Networks (DQN). Since the model integrates with DRL, it can achieve high post-detection accuracy. The system can also demonstrate robustness against adversarial attacks, with minimal degradation in accuracy, further enhanced by a denoising autoencoder (DAE) integrated with the IDS. Sami et al. [115] developed a Phishing Email Detection System (PEDS), which is an online phishing email detection framework based on both supervised and unsupervised learning techniques and can dynamically adapt to the changes of new phishing emails. Here, the supervised technique is used to develop a detection model by training with a dataset, and the unsupervised technique tries to adapt the detection model by using a newly delivered email to the system. The model utilizes NN, RL, data mining classification, and a group of algorithms to detect phishing attacks, and it can explore new phishing behaviours. Initially, it extracts 50 features from the offline dataset and then chooses the most important features by using the FEaR algorithm. The Dynamic Evolving Neural Network, using the Reinforcement Learning (DENNuRL) algorithm, then generates the PEDS for online email classification. Finally, RL is used to continuously adapt PEDS with the newly explored behaviours online. This generated PEDS is a trained NN that can detect ham and phishing emails online, even though the first NN was generated using the offline dataset. The PEDS is then continuously adopted by the RL agent with the reflection of changes in the online dataset.

4.6.3. External facilitation

Detecting anomalies in IoT is a challenge, since a wide variety of devices are being introduced daily, and hence, there is a dynamic landscape of threats for those devices. For instance, these devices have limited computational resources. Moreover, the false alarm rate in IoT anomaly detection is also high because of the heterogeneity of the devices. Additionally, IoT devices generate very little traffic. Nguyen et al. [74] introduced DÏoT, which is an FL-based anomaly detection method. It is a self-learning distributed security monitoring approach utilizing device-type-specific anomaly detection models. It can operate without any human intervention and labelled data. It automatically detects the type of devices in the network quickly with high accuracy through the AuDI [149] approach and prepares models with each device type based on their behaviour. Fig. 13 shows the DÏoT system, which consists of a Security Gateway to monitor the devices and detect anomalies from compromised devices. An IoT Security Service supports this security gateway by maintaining a repository of device-type-specific anomaly detection models. The security gateway checks the device type when a new device is added to the system. It gets the anomaly detection models for that device from the IoT security service support. The anomaly detection models are trained with FL models, where the security gateway utilizes the locally collected data; whereas, the IoT security service supports aggregating them to a global model. The authors used the feature selection method and Gated Recurrent Units (GRUs) algorithm to ensure adequate training even with very little training data. The result shows that DIOT can detect unknown attacks with very high detection accuracy and very fast. It can detect a compromised device infected with the infamous Mirai malware with a 95.6% detection rate in 257 ms.

4.6.4. Key takeaways

Modern ML-based network security is increasingly leaning towards automation to reduce human intervention. The key approaches can be grouped into three areas: data processing, model optimization, and external facilitation. Among them, model optimization received the most focus among the presented solutions. It includes the use of AutoML for model selection and tuning, automated parameter selection methods, unsupervised learning strategies, and context adaptation mechanisms. The data processing technique, such as active learning, can minimize manual labelling, and AutoML can perform data preprocessing and

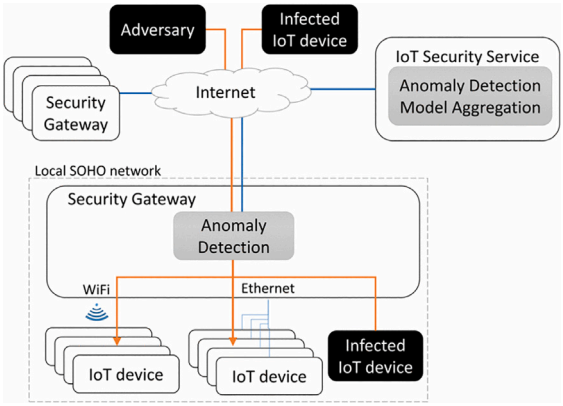


Fig. 13. DÏoT system model [74].

feature engineering. External facilitation received the least focus, addressing how the combination of device-type-specific detection, FL, and unsupervised learning can enable IoT anomaly detection with minimal human input. ML-based network security solutions can largely manage themselves when augmented with these techniques, while they can be more efficient in terms of performance and resource cost, scalable, and quick to respond to new threats. This not only reduces the reliance on human expertise but also reduces the scope of human errors. However, it is worth mentioning that completely removing humans from the loop should not be the goal, since it will introduce risks and ethical concerns.

4.7. Resource and performance efficiency trade-offs

Achieving resource or performance efficiency is not straightforward, and a tradeoff between resource efficiency and performance according to the use-case or implementation scenario is always sought. The trade-off is very clear in resource-constrained environments, and the choice of the best ML algorithm to run in a resource-constrained environment leads to competing optimization targets. In many cases, a solution may show high accuracy and latency with the price of higher memory usage and energy consumption. For instance, Camélia et al. [150] characterized different IDS model implementations in CPUs and GPUs for swarms of drone test cases. The authors tested RF, CNN, and DNN (1D and 2D) algorithms with different feature set setups with the UNSW-NB15 data set. They are (i) without any feature selection, 197 features in total, (ii) 16 features selected with expert selection [151], and (iii) 25 features selected with VAE. It resulted in 24 implementations to examine four IDS models combined with these three types of feature selections in two platforms (CPU and GPU). The authors found that the RF model sizes are the biggest ones on both platforms, while Auto-Encoder RF (AE-RF) was too big to be compiled for the Raspberry PI4 CPU, and thus, it was left out of the process. This shows that the size of the model is a key factor to consider for devices with limited resources. The AE-RF model for the GPU was much bigger than other models. Here, Emlearn was used to convert the model for CPU and HummingBird.ml for GPU use-cases. The fastest model was the RF with expert feature selection (ES-RF) on the Nvidia Xavier GPU (235 ms). The slowest was the CNN without feature selection (NoFS-CNN) on the Raspberry Pi (41 min 38 s), a 10,000x difference. For most models, the GPU was faster than the CPU. However, for DNN1 and CNN with AE, the CPU was faster, mainly due to the data transfer between the CPU and GPU. Sequential data transfer resulted in large memory overhead but better per-packet reactivity. When the model size is small, it cannot utilize the benefit of the parallelism of GPU, which is what happened with DNN1 and CNN here. Therefore, in detecting intrusions, the complexity of the overall environment, the amount of data, and the deployment platforms must be accounted for when selecting ML models.

On the side of latency, GPU is faster than CPU in most cases, however, the energy consumption is higher, as studied in [150]. CPU, on the other hand, can be better for shallow ML models. Shallow models can be used in IoT environments having CPUs, whereas GPUs are better in centralized servers or server farms since the threat landscape is bigger and the assets to be secured are comparatively costly. Furthermore, the data rates also impacts the selection of ML models, and the resource consumption footprints. GPUs are better for higher data rates for faster processing, but with a higher energy footprint. For security analysis or anomaly detection in the network layer, routers or switches, the data rate is very high and the devices must work at line rate. Therefore, there are strict latency implications, meaning that energy consumption must be sacrificed if we deploy ML for anomaly detection. Therefore, GPUs will be the preferred choice in the network layer.

There is also a direct performance trade-off between resources and efficiency of ML models. For instance, model pruning and quantization, and data sizes for training and inference impact the accuracy of ML models. For critical applications accuracy and model efficiency cannot be compromised for resource efficiency. Similarly, bandwidth and transceiver consumption can be ignored for time-sensitive applications. In privacy-sensitive applications, techniques such as homomorphic encryption would be needed that require higher computation not only for ML models but also due to saving sensitive information during the operations of ML. Therefore, there will always exist a trade-off among different objectives of security and resource-efficiency, particularly in communications networks, where the situation will remain dynamic compared to other fields and application areas of ML.

5. Lesson learned

Proper utilization of processing, energy, storage, spectrum and other resources is very crucial in a communication network due to their scarcity and cost. For instance, the wireless spectrum is a limited and extremely expensive resource. Hence, several initiatives are in place to fairly share the available spectrum with novel technologies or use it intelligently. Hence, huge quantities of raw data acquisition for training of large ML models can be costly. Similarly, processing, storage and battery capabilities in most end-user devices are limited. Therefore, local information processing at the end-user devices may be limited. Consequently, operating large ML model-based security applications often poses significant challenges, which render operations either impossible or resource-intensive for these devices. Furthermore, communications systems must deliver data with minimal delay to enable real-time applications and provide a satisfactory level of QoS. Hence, ML-based operations should be extremely fast to comply with strict latency requirements. Lastly, the exponential growth of network traffic data enables high-accuracy deep learning models for advanced security services. However, these models have higher algorithm complexity and significant energy consumption, which also raises environmental concerns.

This work studied a wide range of ML-based network security solutions to investigate the resource consumption of different ML models and techniques. Our analysis revealed that time or latency is the most emphasized KPI, receiving the majority of research focus, as shown in Fig. 14, followed by processing, memory, and energy. In contrast, human resource and bandwidth received comparatively less attention, with bandwidth being the least studied. We observed that supervised learning techniques are very commonly used for minimizing energy, processing, memory, bandwidth, and time consumption. We also observed that techniques to improve efficiency often cross these resource dimensions. For example, processing is a major contributor to energy consumption; thus, reducing the use of processing will reduce energy consumption. Furthermore, unsupervised learning, supervised learning, RL, AutoML, and active learning are commonly used to reduce human resource requirements. Table 10 shows the most important supervised, unsupervised, and reinforcement learning techniques from

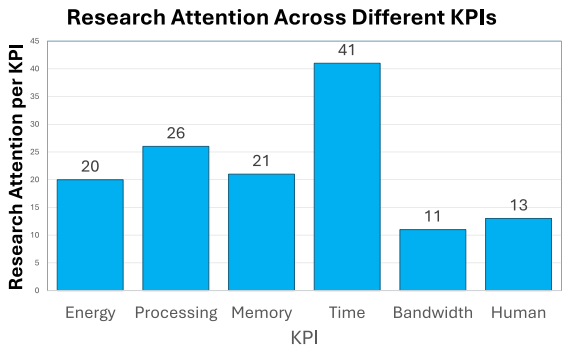


Fig. 14. Research attention across different KPIs.

the investigated solutions that performed well in terms of reducing the corresponding resources. Various tools that can be used to measure the resource consumption of ML are also listed in Table 11.

Various resource-aware solutions have been developed in network security by employing different techniques to improve resource efficiency, as discussed throughout this article. We have categorized these techniques into three types: data processing, model optimization, and external facilitation. The lessons learned in each category are described below.

5.1. Data processing

In network security, it is common practice to minimize resource usage by efficiently processing input data before applying machine learning algorithms. High-dimensional data often contains noise and irrelevant features, making it crucial to pre-process and select the right features. Additionally, the size of the dataset is directly related to the processing resources required; larger datasets demand more resources compared to smaller ones. By utilizing various data processing techniques, resource efficiency can be achieved by making the data more manageable and easy to process. Feature engineering is one of the common data-handling techniques employed. It normally pre-processes network data before running the ML model application for security purposes, including IDS, DDoS detection, and anomaly detection. By generating, selecting, and extracting useful features from live traffic data, it can reduce resource usage significantly, while enabling network security. At the pre-processing stage, feature selection, feature extraction, and traffic reduction are also common techniques in network security. They are mainly applied to reduce the data dimensionality and, consequently, resource usage. Training ML models with a small subset of data is another resource-efficient practice. However, reducing the size of the dataset and the number of features may achieve reduction in energy consumption, but it may reduce the accuracy too [79]. Hence, a balanced trade-off should be considered based on the needs of the application. Additionally, preparing smaller datasets is challenging and requires very careful selection and statistical computations. Packet batching can reduce the system memory and CPU usage significantly, however, there might be a small reduction in model accuracy. The packet batching can be employed in botnet detection, flow detection, IDS, DDoS detection, and other applications of network security. Furthermore, integrating big data processing methods with ML-based network security at the data processing stage can enhance resource efficiency and achieve online detection with better accuracy and scalability. Finally, by automating different parts of data processing, such as data labelling, pre-processing, feature engineering, human intervention in data processing can be reduced significantly.

5.2. Model optimization

Optimization of an ML model is another way of reducing the resource usage of ML-based network security. In several solutions, resource efficiency was achieved through different model optimization methods. Among them, distributed computation is a valuable technique to reduce energy consumption. In this context, ML computations are distributed among various devices in the cloud–edge continuum based on the use case and application needs. Distributing network traffic across multiple independent solutions, like NIDS, is an effective way to scale up network security by balancing workloads. To optimize the performance of network security appliances with multiple instances, various load balancing factors are considered. However, certain load-balancing approaches, such as round-robin, may not be suitable for certain ML-based network security, like NIDS, and may degrade the overall ML performance [152]. Another approach to resource optimization is selecting the best-performing machine learning algorithms for network security through a comprehensive analysis and comparison of performance and resource consumption. By employing suitably tailored ML algorithms, various resource usage can be reduced significantly. However, even though a model may perform very well in terms of resource consumption, it may not always be the best choice since its performance may be lacking, for example, in [86], NB performed the best in terms of energy consumption, but lowest in terms of accuracy. Therefore, the trade-off between resources and energy must be considered, and alternative models that require more energy but offer greater accuracy should also be evaluated. Distributed or collaborative learning can be effective in protecting the network of resource-constraint devices. It can be employed in developing an in-network security architecture, where the distributed switches can work together to learn a unified attack detection policy via a centralized, synchronized platform and, hence, reduce memory and other resource consumption. However, it may not always be true, since it can face various challenges, such as poor connectivity, device heterogeneity, high latency, and security overhead. By leveraging the existing knowledge of a pre-trained model, a significant amount of time and resources can be saved, since it does not require training from scratch. However, pre-trained models can have complex architectures. This may limit their deployment in real use cases due to the lack of interpretability by practitioners, who may prefer to customize other alternatives to the specific needs while integrating in the system [153]. Different model compression techniques, like quantization and pruning, can aid in the reduction of energy, processing, and memory consumption. Furthermore, automating parameter tuning, pre-processing, and context adaptation can minimize human resources in network security.

5.3. External facilitation

Different external factors can aid ML in reducing resource consumption. For instance, computation offloading can be implemented to reduce energy and time consumption by transferring complex tasks to a more powerful processing unit in network security, like IDS for UAVs. Dedicated hardware designed for a specific ML-based network security application can reduce energy, processing, and memory consumption. Integration of heuristic rules with ML can significantly reduce the workload of ML in network security, consequently reducing memory, CPU, and energy consumption. However, the practitioner must have an extensive understanding of the context. Lightweight IDS can be developed for IoT gateways by reducing the number of incoming input size using a per-host detection unit, resulting in memory and execution time reduction. The implementation of a Hadoop Framework, like Spark with ML, enables ML in early anomaly detection, consequently reducing resource usage. By using ML with specialized network architecture, like SDN, abnormal traffic can be identified efficiently, and as a result, a significant amount of bandwidth can be saved. Here, the SDN controller can aid with the statistical information of the overall

Table 10
Top ML algorithms for network security solutions: Resource usage and detection performance analysis.

ML types	Energy	Processing	Memory	Time	Bandwidth	Human resource
Supervised	LSTM RF MLP DT FL(MLP) NB Hybrid AdaBoost DNN	RF SVM DT NB LR XGB DNN LSTM	MLP kNN DT DNN Hybrid RF NB XGB Custom	FL (LR) DT MLP LR XGB Hybrid NB RF SVM ELM	RF DT DL	GPR LGBM GPR RF DT Custom
Unsupervised	–	FL(Hybrid) VAE	FL(Hybrid)	FL(DRL) ANN KNN	FL (Hybrid)	GA LSTM K-Means
Reinforcement learning	–	–	–	Custom	DQL	DRL Custom

Table 11
Tools to measure consumption of resources.

Energy	Processing	Memory	Time	Bandwidth
Tegrastats [72] Makerhawk UM25C USB Tester [73] RM Stream Line [154] Powmon [155] McPAT [156] PAPI [157] eco2AI [9] Altera's Power Monitor tool [105] Custom power measurement platform [105]	Tegrastats [72] Conky [71] Linux command "top" [106] Resmon [158] Simple System Monitor [65]	Tegrastats [72] Conky [71] memory-profiler [87] Resmon [158] Simple System Monitor [65] tracemalloc [106] Fil memory profiler [159] Sciagraph profiler [160]	Tegrastats [72] ipython-autotimer [87] Conky [71] time [161]	Bmon [72,73] iftop [71]

network traffic and block the source of the abnormal traffic after detection with ML. Some queuing algorithms, like Cyclic queuing, can also help ML to achieve resource consumption reduction for resource-limited edge networks by narrowing down the suspected flow and utilizing the history of the queuing size around the stage transition. Optimal resource allocation among the trusted devices with ML can also reduce bandwidth consumption. Specialized hardware, such as DVFS, can help reduce resource consumption. Modern CPUs often use DVFS to dynamically adjust their frequency, improving energy efficiency. Finally, automatic device identification and the use of device-type-specific self-learning network security models can reduce the need for human resources.

6. Challenges and future research directions

Achieving resource efficiency in ML-based network security solutions is not straightforward due to various challenges. In this section, we discuss the most important challenges and research gaps that need further research.

6.1. Data volume

The volume and heterogeneity of total network traffic are continually increasing, presenting a significant challenge for many ML-based solutions. As the volume of data grows, it becomes increasingly difficult to resource-efficiently manage data, mine useful information, visualize, and make decisions [162]. Larger datasets require more resources to process and analyse, and this translates to longer training times and expanded hardware demands. Larger and heterogeneous data sets may lead to more complex and computationally demanding ML models, with many layers and parameters. Furthermore, big data introduces architectural challenges related to scalability and latency. Cloud-based solutions can provide scalable resources for handling large datasets, and distributed computing paradigms may accelerate training. Applications that rely on inference on resource-constrained devices will struggle with increasing real-time data flows. ML-based security applications need to apply pre-processing techniques, including data cleaning, normalization, and feature engineering, to minimize the computational burden as well as model optimizations to make models smaller and faster. For many security applications, future work is needed to understand how these mitigations impact ML-model efficiency to achieve their security goals.

6.2. Dynamic environments and changing security landscapes

One of the significant challenges encountered during training is that complex architectures demand higher computational resources [163]. It is also essential to retrain a deployed ML model to deal with concept drift [164], which refers to the continuous data changes for various factors, such as system modification, user preference changes, and seasonality. Further, adversaries are also constantly evolving and developing new attacks, which circumvent outdated ML-based security solutions. If the model is not updated, it may cost from the result of degraded performance and missed threats. On the other hand, training a model with fresh data requires resources. Hence, not retraining often enough and retraining too often both have different resource costs [165]. Additionally, retraining distributed ML models often demands higher latency, especially for resource-constrained devices. Hence, it may require a significant amount of retraining time. Furthermore, adversarial attacks can also occur during the training time. Therefore, it is important to carefully select the data for training to avoid model corruption. Moreover, when a trained model is retrained with new data, its size and complexity may increase significantly. If a resource-efficient network security system is deployed on a resource-constrained device without considering this fact, the solution may be jeopardized due to the increased size and complexity achieved through retraining with new data.

6.3. Hardware constraints and lightweight ML

Hardware dictates what kind of ML and or ML-based security applications are possible. Resource-constrained devices, like IoT, that have limited processing, battery, memory, and other resources, can restrict them from running complex ML algorithms [166]. As the data volume increases, the computation, for instance, for training, will increase. In resource-constrained environments, the hardware capacities cannot be increased at run-time as the data volume increases. Furthermore, the heterogeneity of IoT devices complicates the development of a uniform, resource-efficient solution. Additionally, striking a careful balance between the trade-off of security, performance, and resource usage is also challenging for resource-constrained and heterogeneous devices. Consequently, to support communication architectures involving restricted hardware platforms, such as IoT networks, and edge computing paradigms, there is a need to develop lightweight ML approaches, such as TinyML [167]. Security research challenges related to TinyML include, e.g., vulnerabilities of minimal ML models against adversarial input data [168].

6.4. Quantum computing

Quantum computers can process certain types of data very faster than traditional computers, and thus there are various proposals for quantum machine learning [169–171]. Generally, quantum computing can help in speeding up the computations needed in training ML models, sampling data, feature extraction, encoding, dimensionality reduction, and improving overall ML optimization. However, the research on quantum computing in general and quantum machine learning in particular is still in its early phase, making its practical use highly challenging. Furthermore, the current quantum computers require huge amounts of energy, for instance, for the cooling environment of cryogenics to maintain low temperatures, error correction, and its control electronics. Quantum computing have increasingly high potential in encryption in the cybersecurity space, however, still more research is needed in the use of ML for cybersecurity using quantum computers in resource-efficient manner.

6.5. Resource trade-offs between security and peacetime applications

A significant challenge in optimizing resource efficiency for network security solutions is the fact that security is not the primary function of a communication system. Therefore, it must coexist with the system's other applications without causing disruptions. Moreover, many systems have duties beyond processing applications. For instance, in the case of aerial drones, the majority of their energy is used for movement, followed by processing tasks [172]. Consequently, careful resource management is crucial to achieve resource efficiency for any network security solution, including ML-based security. Future work is needed to develop dual-use technologies that demonstrate applications achieving both security and other goals at the same time and with the same resources, i.e., approaches which solve "peacetime" problems, while looking at security threats and, e.g., hostility-related anomalies. Furthermore, resource consumption translates to the economic and environmental dimensions of sustainability. Therefore, more research is needed on the sustainability of network and cybersecurity technologies that use ML approaches.

6.6. Trustworthiness and privacy aspects

ML also has some fragility that makes it vulnerable to various attacks [173,174]. The training data may be poisoned, which leads to backdoors in the models, and the input data may be crafted to evade detection and mislead decision-making. The attackers can achieve model extraction, training data extraction, and misclassification by altering the dataset with active attacks like poisoning attacks or without changing the dataset with passive attacks, such as reverse attack, adversarial attack, and membership inference attack [175]. A user involved in a training process may also violate privacy, for instance, user behaviour can reveal sensitive data, learned models, or other important parameters.

Approaches for increasing the trustworthiness and robustness of ML have impact on resource consumption:

- Adversarial training approaches [176] train ML models using attack data as well, which increases training time and computational resources.
- Ensemble methods [177], which combine multiple models to improve the reliability of results, increase memory and computational costs both for training and inference time.
- Pre-processing schemes generating additional training samples or noise, such as data augmentation and differential privacy, increase storage requirements and computation overhead during training.
- Monitoring schemes such as outlier detection in pre-processing and anomaly detection over model's performance, can be computationally expensive, especially for large data volumes.
- Some approaches increasing explainability of ML [178], including Shapley additive explanation (SHAP) and counterfactual explanations, can be computationally expensive, while approaches replacing complex unexplainable neural networks with simpler and more easy-to-explain alternatives, such as decision trees, can be more efficient, at least with smaller datasets.

Preserving privacy is another key concern and a major challenge for ML, which also includes resource-efficient solutions. ML is utilized to analyse and harvest a huge amount of data while optimizing networks, such as self-organizing networks, which can be a potential privacy concern for the users [179]. Approaches to increasing privacy include data minimization and collecting only necessary data, which can have a positive impact on the reduction of storage and computing resources. Furthermore, techniques like differential privacy, anonymization, and federated learning can be applied at the expense of increased computational and communications overhead. Other cryptographic schemes, such as homomorphic encryption and secure multi-party computation, significantly increase computational complexity, however increase privacy.

6.7. Ethical aspects

The integration of AI/ML into cybersecurity introduces several ethical concerns that require careful consideration. Generally, when it comes to cybersecurity, ethical considerations are given less weight. This can be under the guise of accountability, where data privacy concerns are ignored. Similarly, the environmental sustainability aspects have not received enough research attention, which can be seen from very less active research output on the topic. On the application of ML in network or cybersecurity, there are technical challenges that need further research. For example, ML systems can make biased decisions. To address this issue, bias detection and mitigation techniques should be employed, including the use of diverse and representative datasets. Another major concern is the lack of transparency, which can lead to ambiguity in decision-making and reduced trust in AI or ML systems. Explainable AI (XAI) can help mitigate this problem, but it often increases model complexity and may involve performance trade-offs. Additional challenges include determining accountability and liability for the system's decisions, as well as ensuring appropriate human oversight to maintain ethical and effective operation [180]. Addressing these concerns can hinder the development of resource-efficient cybersecurity systems for future networks, as they require additional features, expert involvement, energy, and other resources. A promising direction for future research is to develop adaptive frameworks for network security that can dynamically balance ethical considerations and resource constraints based on contextual priorities.

7. Conclusions

The applications of ML in the security of communications networks have become inevitable due to the increasing complexity of communications networks and exponential growth in traffic volumes. The applications of ML, however, require dedicated resources for operations of ML, ranging from gathering raw data over communication channels, to processing the data either in centralized servers or distributed computing nodes, and then distributing intelligent, actionable decisions. This article studies the consumption of important network resources, which include energy, processing, memory, time, bandwidth, and human involvement. Existing research that improves network security while achieving efficiency in terms of these resources is investigated. A wide range of ML-based network security solutions are studied where resource efficiency is achieved through data processing, model optimization, and external facilitation. It is important to note that most research is concentrated on improving latency or time consumption of ML techniques, which itself is highly dependent on the processing capabilities of the systems. Moreover, research on minimizing the consumption of bandwidth, which translates to frequency occupancy in wireless networks, has received less research attention comparatively. In this vein, this article highlights the most important research challenges that need further research to address the most important challenges that arise with the deployment of ML in communications networks. Since resource consumption translates to costs and sustainability, more research is needed on the implications of ML-based network security on costs and sustainability. In summary, this article elaborates the existing trends and future research directions in a first attempt to study the resource consumption of ML-based network security techniques and procedures.

CRediT authorship contribution statement

Md Muzammal Hoque: Writing – original draft, Methodology, Investigation, Formal analysis. **Ijaz Ahmad:** Writing – original draft, Supervision, Resources, Project administration, Methodology, Conceptualization. **Jani Suomalainen:** Writing – original draft, Validation, Investigation, Formal analysis, Conceptualization. **Paolo Dini:** Writing – original draft, Validation, Methodology, Formal analysis. **Mohammad Tahir:** Writing – review & editing, Writing – original draft, Supervision, Methodology, Investigation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This research is funded by the Business Finland project called SUNSET-6G (grant 8682/31/2022).

Data availability

No data was used for the research described in the article.

References

[1] C.-X. Wang, X. You, X. Gao, X. Zhu, Z. Li, C. Zhang, H. Wang, Y. Huang, Y. Chen, H. Haas, J.S. Thompson, E.G. Larsson, M.D. Renzo, W. Tong, P. Zhu, X. Shen, H.V. Poor, L. Hanzo, On the road to 6G: Visions, requirements, key technologies, and testbeds, IEEE Commun. Surv. Tutor. 25 (2) (2023) 905–974.

[2] I. Ahmad, S. Shahabuddin, H. Malik, E. Harjula, T. Leppänen, L. Lovén, A. Anttonen, A.H. Sodhro, M. Mahtab Alam, M. Juntti, A. Ylä-Jääski, T. Sauter, A. Gurtov, M. Ylianttila, J. Riekki, Machine learning meets communication networks: Current trends and future challenges, IEEE Access 8 (2020) 223418–223460.

[3] Y. Sun, M. Peng, Y. Zhou, Y. Huang, S. Mao, Application of machine learning in wireless networks: Key techniques and open issues, IEEE Commun. Surv. Tutor. 21 (4) (2019) 3072–3108.

[4] P.V. Klaine, M.A. Imran, O. Onireti, R.D. Souza, A survey of machine learning techniques applied to Self-Organizing cellular networks, IEEE Commun. Surv. Tutor. 19 (4) (2017) 2392–2431.

[5] S. Ali, W. Saad, N. Rajatheva, K. Chang, D. Steinbach, B. Sliwa, C. Wietfeld, K. Mei, H. Shiri, H.-J. Zepernick, et al., 6G white paper on machine learning in wireless communication networks, 2020, arXiv preprint arXiv:2004.13875.

[6] L. Von Ahn, M. Blum, N.J. Hopper, J. Langford, CAPTCHA: Using hard AI problems for security, in: Advances in Cryptology—EUROCRYPT 2003: International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4–8, 2003 Proceedings 22, Springer, 2003, pp. 294–311.

[7] E. Rodríguez, B. Otero, N. Gutiérrez, R. Canal, A survey of deep learning techniques for cybersecurity in mobile networks, IEEE Commun. Surv. Tutor. 23 (3) (2021) 1920–1955.

[8] T. Nguyen, H. Nguyen, A. Ijaz, S. Sheikhi, A.V. Vasilakos, P. Kostakos, Large language models in 6G security: challenges and opportunities, 2024, arXiv preprint arXiv:2403.12239.

[9] S.A. Budennyy, V.D. Lazarev, N.N. Zakharenko, A.N. Korovin, O.A. Plosskaya, D.V. Dimitrov, V.S. Akhripkin, I.V. Pavlov, I.V. Oseledets, I.S. Barsola, I.V. Egorov, A.A. Kosterina, L.E. Zhukov, eco2AI: Carbon emissions tracking of machine learning models as the first step towards sustainable AI, Dokl. Math. 106 (1) (2022) S118–S128.

[10] A.E. Brownlee, J. Adair, S.O. Haraldsson, J. Jabbo, Exploring the accuracy – Energy Trade-off in machine learning, in: 2021 IEEE/ACM International Workshop on Genetic Improvement, GI, 2021, pp. 11–18.

[11] A transparent and standards-based way to assess the environmental impact of AI systems, 2024, pp. 1–15, URL https://onestore.nokia.com/asset/214115.

[12] R. Schwartz, J. Dodge, N.A. Smith, O. Etzioni, Green AI, Commun. ACM 63 (12) (2020) 54–63.

[13] I. Ahmad, S. Shahabuddin, T. Kumar, J. Okwuibe, A. Gurtov, M. Ylianttila, Security for 5G and beyond, IEEE Commun. Surv. Tutor. 21 (4) (2019) 3682–3722.

[14] M. Husák, J. Komárková, E. Bou-Harb, P. Čeleda, Survey of attack projection, prediction, and forecasting in cyber security, IEEE Commun. Surv. Tutor. 21 (1) (2019) 640–660.

[15] K. Shaukat, S. Luo, V. Varadharajan, I.A. Hameed, M. Xu, A survey on machine learning techniques for cyber security in the last decade, IEEE Access 8 (2020) 222310–222354.

[16] X. Lin, An overview of 5G advanced evolution in 3GPP release 18, IEEE Commun. Stand. Mag. 6 (3) (2022) 77–83.

[17] T.S.S.O. ITU, Security architecture for systems providing end-to-end communications, URL https://www.itu.int/rec/T-REC-X.805-200310-I/en.

[18] I. Ahmad, S. Namal, M. Ylianttila, A. Gurtov, Security in software defined networks: A survey, IEEE Commun. Surv. Tutor. 17 (4) (2015) 2317–2346.

[19] P. Chemouil, P. Hui, W. Kellerer, Y. Li, R. Stadler, D. Tao, Y. Wen, Y. Zhang, Special issue on artificial intelligence and machine learning for networking and communications, IEEE J. Sel. Areas Commun. 37 (6) (2019) 1185–1191.

[20] E. Baccour, N. Mhaisen, A.A. Abdellatif, A. Erbad, A. Mohamed, M. Hamdi, M. Guizani, Pervasive AI for IoT applications: A survey on Resource-Efficient distributed artificial intelligence, IEEE Commun. Surv. Tutor. 24 (4) (2022) 2366–2418.

[21] G. Menghani, Efficient deep learning: A survey on making deep learning models smaller, faster, and better, ACM Comput. Surv. 55 (12) (2023).

[22] A. Gouissem, Z. Chkirbene, R. Hamila, A comprehensive survey on energy efficiency in federated learning: Strategies and challenges, in: 2024 IEEE 8th Energy Conference, ENERGYCON, 2024, pp. 1–6.

[23] Z.M. Fadlullah, B. Mao, N. Kato, Balancing QoS and security in the edge: Existing practices, challenges, and 6G opportunities with machine learning, IEEE Commun. Surv. Tutor. (2022) 1–1.

[24] J. Lee, L. Mukhanov, A.S. Molahosseini, U. Minhas, Y. Hua, J. Martinez del Rincon, K. Dichev, C.-H. Hong, H. Vandierendonck, Resource-Efficient convolutional networks: A survey on model-, arithmetic-, and implementation-level techniques, ACM Comput. Surv. 55 (13s) (2023).

[25] S. Bavikadi, P.R. Sutradhar, K.N. Khasawneh, A. Ganguly, S.M. Pudukotai Dinakarrao, A review of In-Memory computing architectures for machine learning applications, in: Proceedings of the 2020 on Great Lakes Symposium on VLSI, GLSVLSI '20, Association for Computing Machinery, New York, NY, USA, 2020, pp. 89–94.

[26] Y. Shi, K. Yang, T. Jiang, J. Zhang, K.B. Letaief, Communication-Efficient edge AI: Algorithms and systems, IEEE Commun. Surv. Tutor. 22 (4) (2020) 2167–2191.

[27] E. Barbierato, A. Gatti, Toward green AI: A methodological survey of the scientific literature, IEEE Access 12 (2024) 23989–24013.

[28] D. Katare, D. Perino, J. Nurmi, M. Warnier, M. Janssen, A.Y. Ding, A survey on approximate edge AI for energy efficient autonomous driving services, IEEE Commun. Surv. Tutor. 25 (4) (2023) 2714–2754.

[29] A. Mughees, M. Tahir, M.A. Sheikh, A. Ahad, Towards energy efficient 5G networks using machine learning: Taxonomy, research challenges, and future research directions, IEEE Access 8 (2020) 187498–187522.

[30] H.-I. Liu, M. Galindo, H. Xie, L.-K. Wong, H.-H. Shuai, Y.-H. Li, W.-H. Cheng, Lightweight deep learning for resource-constrained environments: A survey, ACM Comput. Surv. 56 (10) (2024).

[31] Z. Ning, H. Hu, X. Wang, L. Guo, S. Guo, G. Wang, X. Gao, Mobile edge computing and machine learning in the internet of unmanned aerial vehicles: A survey, ACM Comput. Surv. 56 (1) (2023).

[32] Z. Ye, W. Gao, Q. Hu, P. Sun, X. Wang, Y. Luo, T. Zhang, Y. Wen, Deep learning workload scheduling in GPU datacenters: A survey, ACM Comput. Surv. 56 (6) (2024).

[33] J. Borrego-Carazo, D. Castells-Rufas, E. Biempica, J. Carrabina, Resource-Constrained machine learning for ADAS: A systematic review, IEEE Access 8 (2020) 40573–40598.

[34] S.S. Panwar, M.M.S. Rauthan, V. Barthwal, A systematic review on effective energy utilization management strategies in cloud data centers, J. Cloud Comput. 11 (1) (2022) 95.

[35] M. Mahamat, G. Jaber, A. Bouabdallah, Achieving efficient energy-aware security in IoT networks: a survey of recent solutions and research challenges, Wirel. Netw. 29 (2) (2023) 787–808.

[36] X. Wu, L. Xiao, Y. Sun, J. Zhang, T. Ma, L. He, A survey of human-in-the-loop for machine learning, Future Gener. Comput. Syst. 135 (2022) 364–381.

[37] E. Mosqueira-Rey, E. Hernández-Pereira, D. Alonso-Ríos, J. Bobes-Bascarán, Á. Fernández-Leal, Human-in-the-loop machine learning: a state of the art, Artif. Intell. Rev. 56 (4) (2023) 3005–3054.

[38] P. Mishra, V. Varadharajan, U. Tupakula, E.S. Pilli, A detailed investigation and analysis of using machine learning techniques for intrusion detection, IEEE Commun. Surv. Tutor. 21 (1) (2019) 686–728.

[39] T.M. Hoang, A. Vahid, H.D. Tuan, L. Hanzo, Physical layer authentication and security design in the machine learning era, IEEE Commun. Surv. Tutor. 26 (3) (2024) 1830–1860.

[40] T. Kumar, J. Partala, T. Nguyen, L. Agrawal, A. Mondal, A. Kumar, I. Ahmad, E. Peltonen, S. Pirttikangas, E. Harjula, Secure edge intelligence in the 6G era, Secur. Priv. 6G Massive IoT (2025) 35–52.

[41] I. Ahmad, S. Shahabuddin, T. Sauter, E. Harjula, T. Kumar, M. Meisel, M. Juntti, M. Ylianttila, The challenges of artificial intelligence in wireless networks for the Internet of Things: Exploring opportunities for growth, IEEE Ind. Electron. Mag. 15 (1) (2021) 16–29.

[42] D. Rosendo, A. Costan, P. Valduriez, G. Antoniu, Distributed intelligence on the edge-to-cloud continuum: A systematic literature review, J. Parallel Distrib. Comput. 166 (2022) 71–94.

[43] R. Singh, S.K. Singh, S. Kumar, S.S. Gill, SDN-aided edge computing-enabled AI for IoT and smart cities, in: SDN-Supported Edge-Cloud Interplay for Next Generation Internet of Things, Chapman and Hall/CRC, 2022, pp. 41–70.

[44] A. Ahmad, E. Harjula, M. Ylianttila, I. Ahmad, Evaluation of machine learning techniques for security in SDN, in: 2020 IEEE Globecom Workshops (GC Wkshps), 2020, pp. 1–6, http://dx.doi.org/10.1109/GCWkshps50303.2020.9367477.

[45] J. Gil Herrera, J.F. Botero, Resource allocation in NFV: A comprehensive survey, IEEE Trans. Netw. Serv. Manag. 13 (3) (2016) 518–532, http://dx.doi.org/10.1109/TNSM.2016.2598420.

[46] H. Babbar, S. Rani, S.A. AlQahtani, Intelligent edge load migration in SDN-IIoT for smart healthcare, IEEE Trans. Ind. Inform. 18 (11) (2022) 8058–8064, http://dx.doi.org/10.1109/TII.2022.3172489.

[47] I. Ahmad, A. Mämmelä, M.M. Mowla, A. Flizikowski, M.A.B. Abbasi, D. Zelenchuk, M. Sinaie, Sustainability in 6G networks: Vision and directions, in: 2023 IEEE Conference on Standards for Communications and Networking, 2023, pp. 202–208.

[48] M. Gutiérrez, M.Á. Moraga, F. García, Analysing the energy impact of different optimisations for machine learning models, in: 2022 International Conference on ICT for Sustainability (ICT4S), 2022, pp. 46–52.

[49] G.S. Nikolić, B.R. Dimitrijević, T.R. Nikolić, M.K. Stojcev, A survey of three types of processing units: CPU, GPU and TPU, in: 2022 57th International Scientific Conference on Information, Communication and Energy Systems and Technologies, ICEST, 2022, pp. 1–6.

[50] A. Jiang, E.F. Haratsch, Machine learning: Enabling and enabled by advances in storage and memory system, IEEE BITS Inf. Theory Mag. (2023) 1–12.

[51] M. Guimarães, D. Carneiro, G. Palumbo, F. Oliveira, Ó. Oliveira, V. Alves, P. Novais, Predicting model training time to optimize distributed machine learning applications, Electron. 12 (4) (2023).

[52] C. Janiesch, P. Zschech, K. Heinrich, Machine learning and deep learning, Electron. Mark. 31 (3) (2021) 685–695.

[53] D. Comer, Computer Networks and Internets, Prentice Hall, 2008.

[54] S. Amershi, M. Cakmak, W.B. Knox, T. Kulesza, Power to the people: The role of humans in interactive machine learning, AI Mag. 35 (4) (2014) 105–120.

[55] S.K. Karmaker ("Santu"), M.M. Hassan, M.J. Smith, L. Xu, C. Zhai, K. Veeramachaneni, AutoML to date and beyond: Challenges and opportunities, ACM Comput. Surv. 54 (8) (2021).

[56] L. Holmberg, P. Davidsson, P. Linde, A feature space focus in machine teaching, in: 2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), 2020, pp. 1–2.

[57] A.R. Yusof, N.I. Udzir, A. Selamat, H. Hamdan, M.T. Abdullah, Adaptive feature selection for denial of services (DoS) attack, in: 2017 IEEE Conference on Application, Information and Network Security, AINS, 2017, pp. 81–84.

[58] R. Yaegashi, E. Takeshita, Y. Nakayama, Two-Stage DDoS mitigation with variational Auto-Encoder and cyclic queuing, in: ICC 2022 - IEEE International Conference on Communications, 2022, pp. 5421–5426.

[59] S. Mergendahl, J. Li, Rapid: Robust and adaptive detection of distributed Denial-of-Service traffic from the Internet of Things, in: 2020 IEEE Conference on Communications and Network Security, CNS, 2020, pp. 1–9.

[60] T. Visetbunditkun, W. Srichavengsup, DDoS attack detection using ensemble machine learning models with RFE algorithm, in: 2022 7th International Conference on Business and Industrial Research, ICBIR, 2022, pp. 269–273.

[61] Kamaldeep, M. Malik, M. Dutta, Feature engineering and machine learning framework for DDoS attack detection in the standardized Internet of Things, IEEE Internet Things J. 10 (10) (2023) 8658–8669.

[62] P. Bhale, S. Biswas, S. Nandi, ML for IEEE 802.15. 4e/TSCH: Energy efficient approach to detect DDoS attack using machine learning, in: 2021 International Wireless Communications and Mobile Computing, IWCMC, 2021, pp. 1477–1482.

[63] M.I. Kareem, M.N. Jasim, DDOS attack detection using lightweight partial decision tree algorithm, in: 2022 International Conference on Computer Science and Software Engineering, CSASE, 2022, pp. 362–367.

[64] J. Bhayo, S.A. Shah, S. Hameed, A. Ahmed, J. Nasir, D. Draheim, Towards a machine learning-based framework for DDOS attack detection in software-defined IoT (SD-IoT) networks, Eng. Appl. Artif. Intell. 123 (2023) 106432.

[65] S.J. Rani, I. Ioannou, P. Nagaradjane, C. Christophorou, V. Vassiliou, S. Charan, S. Prakash, N. Parekh, A. Pitsillides, Detection of DDoS attacks in D2D communications using machine learning approach, Comput. Commun. 198 (2023) 32–51.

[66] J.J. Kponyo, J.O. Agyemang, G.S. Klogo, J.O. Boateng, Lightweight and host-based denial of service (DoS) detection and defense mechanism for resource-constrained IoT devices, Internet Things 12 (2020) 100319.

[67] R. Al Rahbani, J. Khalife, IoT DDoS traffic detection using adaptive heuristics assisted with machine learning, in: 2022 10th International Symposium on Digital Forensics and Security, ISDFS, 2022, pp. 1–6.

[68] W. He, Y. Liu, H. Yao, T. Mai, N. Zhang, F.R. Yu, Distributed variational Bayes-based In-Network security for the Internet of Things, IEEE Internet Things J. 8 (8) (2021) 6293–6304.

[69] Z. Shi, J. Li, C. Wu, DeepDDoS: Online DDoS attack detection, in: 2019 IEEE Global Communications Conference, GLOBECOM, 2019, pp. 1–6.

[70] E. Gyamfi, A. Jurcut, M-TADS: A multi-trust DoS attack detection system for MEC-enabled industrial IoT, in: 2022 IEEE 27th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks, CAMAD, 2022, pp. 166–172.

[71] J. Jithish, B. Alangot, N. Mahalingam, K.S. Yeo, Distributed anomaly detection in smart grids: A federated learning-based approach, IEEE Access 11 (2023) 7157–7179.

[72] H.T. Truong, B.P. Ta, Q.A. Le, D.M. Nguyen, C.T. Le, H.X. Nguyen, H.T. Do, H.T. Nguyen, K.P. Tran, Light-weight federated learning-based anomaly detection for time-series data in industrial control systems, Comput. Ind. 140 (2022) 103692.

[73] T.T. Huong, T.P. Bac, D.M. Long, T.D. Luong, N.M. Dan, L.A. Quang, L.T. Cong, B.D. Thang, K.P. Tran, Detecting cyberattacks using anomaly detection in industrial control systems: A federated learning approach, Comput. Ind. 132 (2021) 103509.

[74] T.D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, A.-R. Sadeghi, DÏoT: A federated self-learning anomaly detection system for IoT, in: 2019 IEEE 39th International Conference on Distributed Computing Systems, ICDCS, 2019, pp. 756–767.

[75] A. Ahmed, S. Hameed, M. Rafi, Q.K.A. Mirza, An intelligent and time-efficient ddos identification framework for real-time enterprise networks: SAD-F: Spark based anomaly detection framework, IEEE Access 8 (2020) 219483–219502.

[76] A.H. Hamamoto, L.F. Carvalho, L.D.H. Sampaio, T. Abrão, M.L. Proença, Network anomaly detection system using genetic algorithm and fuzzy logic, Expert Syst. Appl. 92 (2018) 390–402.

[77] X. Wang, S. Garg, H. Lin, J. Hu, G. Kaddoum, M. Jalil Piran, M.S. Hossain, Toward accurate anomaly detection in industrial Internet of Things using hierarchical federated learning, IEEE Internet Things J. 9 (10) (2022) 7110–7119.

[78] X.-W. Wu, Y. Cao, R. Dankwa, Pareto-optimal machine learning models for security of IoT applications, in: 2024 International Conference on Smart Applications, Communications and Networking (SmartNets), 2024, pp. 1–6.

[79] J. Corona, M. Antunes, R.L. Aguiar, Eco-friendly intrusion detection: Evaluating energy costs of learning, in: 2023 IEEE 9th World Forum on Internet of Things (WF-IoT), 2023, pp. 1–6.

[80] I. Zakariyya, H. Kalutarage, M.O. Al-Kadri, Towards a robust, effective and resource efficient machine learning technique for IoT security monitoring, Comput. Secur. 133 (2023) 103388.

[81] V. Agate, A. De Paola, S. Drago, P. Ferraro, G.L. Re, Enhancing IoT network security with concept Drift-Aware unsupervised threat detection, in: 2024 IEEE Symposium on Computers and Communications, ISCC, 2024, pp. 1–6.

[82] Y. Katsura, A. Endo, M. Kakiuchi, I. Arai, K. Fujikawa, Lightweight intrusion detection using multiple entropies of traffic behavior in IoT networks, in: 2022 IEEE Global Conference on Artificial Intelligence and Internet of Things (GCAIoT), 2022, pp. 138–145.

[83] L. Buschlinger, R. Rieke, S. Sarda, C. Krauß, Decision Tree-Based rule derivation for intrusion detection in Safety-Critical automotive systems, in: 2022 30th Euromicro International Conference on Parallel, Distributed and Network-Based Processing, PDP, 2022, pp. 246–254.

[84] P.H. Mirzaee, M. Shojafar, H. Bagheri, T.H. Chan, H. Cruickshank, R. Tafazolli, A Two-layer collaborative Vehicle-Edge intrusion detection system for vehicular communications, in: 2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall), 2021, pp. 1–6.

[85] N. Aboueata, S. Alrasbi, A. Erbad, A. Kassler, D. Bhamare, Supervised machine learning techniques for efficient network intrusion detection, in: 2019 28th International Conference on Computer Communication and Networks, ICCCN, 2019, pp. 1–8.

[86] N. Tekin, A. Acar, A. Aris, A.S. Uluagac, V.C. Gungor, Energy consumption of on-device machine learning models for IoT intrusion detection, Internet Things 21 (2023) 100670.

[87] A.-R. Al-Ghuwairi, Y. Sharrab, D. Al-Fraihat, M. AlElaimat, A. Alsarhan, A. Algarni, Intrusion detection in cloud computing based on time series anomalies utilizing machine learning, J. Cloud Comput. 12 (1) (2023) 127.

[88] A.K. Balyan, S. Ahuja, S.K. Sharma, U.K. Lilhore, Machine learning-based intrusion detection system for healthcare data, in: 2022 IEEE VLSI Device Circuit and System (VLSI DCS), 2022, pp. 290–294.

[89] M. Leon, T. Markovic, S. Punnekkat, Comparative evaluation of machine learning algorithms for network intrusion detection and attack classification, in: 2022 International Joint Conference on Neural Networks, IJCNN, 2022, pp. 01–08.

[90] N.Y. Jien, M. Tahir, M. Dabbagh, Y.K. Meng, A. Farooq, Performance evaluation of machine learning algorithms for intrusion detection in IoT applications, in: 2022 IEEE International Conference on Artificial Intelligence in Engineering and Technology, IICAIET, 2022, pp. 1–6.

[91] V.-D. Ngo, T.-C. Vuong, T. Van Luong, H. Tran, Machine learning-based intrusion detection: feature selection versus feature extraction, Clust. Comput. (2023).

[92] U.J. Otokwala, A. Petrovski, Ensemble common features technique for lightweight intrusion detection in industrial control system, in: 2023 IEEE 6th International Conference on Industrial Cyber-Physical Systems, ICPS, 2023, pp. 1–6.

[93] K. Sethi, E. Sai Rupesh, R. Kumar, P. Bera, Y. Venu Madhav, A context-aware robust intrusion detection system: a reinforcement learning-based approach, Int. J. Inf. Secur. 19 (6) (2020) 657–678.

[94] Y.-C. Lai, D. Sudyana, Y.-D. Lin, M. Verkerken, L. D'hooge, T. Wauters, B. Volckaert, F. De Turck, Machine learning based intrusion detection as a service: task assignment and capacity allocation in a multi-tier architecture, in: Proceedings of the 14th IEEE/ACM International Conference on Utility and Cloud Computing Companion, UCC '21, Association for Computing Machinery, New York, NY, USA, 2022.

[95] H. Moudoud, S. Cherkaoui, Empowering security and trust in 5G and beyond: A deep reinforcement learning approach, IEEE Open J. Commun. Soc. 4 (2023) 2410–2420.

[96] S. Sakraoui, A. Ahmim, M. Derdour, M. Ahmim, S. Namane, I. Ben Dhaou, FBMP-IDS: FL-based Blockchain-Powered lightweight MPC-secured IDS for 6G networks, IEEE Access 12 (2024) 105887–105905.

[97] A. Singh, J. Amutha, J. Nagar, S. Sharma, C.-C. Lee, AutoML-ID: automated machine learning model for intrusion detection using wireless sensor network, Sci. Rep. 12 (1) (2022) 9074.

[98] D. Micale, G. Costantino, I. Matteucci, F. Fenzl, R. Rieke, G. Patanè, CAHOOT: a Context-Aware vehicular intrusion detection system, in: 2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), 2022, pp. 1211–1218.

[99] J.K. Samriya, R. Tiwari, X. Cheng, R.K. Singh, A. Shankar, M. Kumar, Network intrusion detection using ACO-DNN model with DVFS based energy optimization in cloud framework, Sustain. Comput.: Inform. Syst. 35 (2022) 100746.

[100] G. De Carvalho Bertoli, L.A. Pereira Júnior, O. Saotome, A.L. Dos Santos, F.A.N. Verri, C.A.C. Marcondes, S. Barbieri, M.S. Rodrigues, J.M. Parente De Oliveira, An End-to-End framework for machine Learning-Based network intrusion detection system, IEEE Access 9 (2021) 106790–106805.

[101] J. Cui, J. Xiao, H. Zhong, J. Zhang, L. Wei, I. Bolodurina, D. He, LH-IDS: Lightweight hybrid intrusion detection system based on differential privacy in VANETs, IEEE Trans. Mob. Comput. 23 (12) (2024) 12195–12210.

[102] H. Xu, Z. Sun, Y. Cao, H. Bilal, A data-driven approach for intrusion and anomaly detection using automated machine learning for the Internet of Things, Soft Comput. 27 (19) (2023) 14469–14481.

[103] A.M. Pasikhani, J.A. Clark, P. Gope, Adversarial RL-based IDS for evolving data environment in 6LoWPAN, IEEE Trans. Inf. Forensics Secur. 17 (2022) 3831–3846.

[104] H. Kawaguchi, Y. Nakatani, S. Okada, IDPS signature classification based on active learning with partial supervision from Network Security Experts, IEEE Access 10 (2022) 105713–105725.

[105] E. Viegas, A.O. Santin, A. França, R. Jasinski, V.A. Pedroni, L.S. Oliveira, Towards an Energy-Efficient Anomaly-Based intrusion detection engine for embedded systems, IEEE Trans. Comput. 66 (1) (2017) 163–177.

[106] H. Gandhi, V. Ribeiro, Packet batching for reducing system resource consumption for botnet detection using network traffic analysis, in: 2022 14th International Conference on COMmunication Systems & NETworkS, COMSNETS, 2022, pp. 1–6.

[107] M. Alauthman, N. Aslam, M. Al-kasassbeh, S. Khan, A. Al-Qerem, K.-K. Raymond Choo, An efficient reinforcement learning-based botnet detection approach, J. Netw. Comput. Appl. 150 (2020) 102479.

[108] J. Holland, P. Schmitt, N. Feamster, P. Mittal, New directions in automated traffic analysis, in: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, CCS '21, Association for Computing Machinery, New York, NY, USA, 2021, pp. 3366–3383.

[109] A. Papanikolaou, A. Alevizopoulos, C. Ilioudis, K. Demertzis, K. Rantos, An autoML network traffic analyzer for cyber threat detection, Int. J. Inf. Secur. 22 (5) (2023) 1511–1530.

[110] A. Karamchandani, A. Mozo, S. Gómez-Canaval, A. Pastor, A methodological framework for optimizing the energy consumption of deep neural networks: a case study of a cyber threat detector, Neural Comput. Appl. 36 (17) (2024) 10297–10338.

[111] D. Jim Solomon Raja, R. Sriranjani, P. Arulmozhi, N. Hemavathi, Unified random forest and hybrid bat optimization based Man-in-the-Middle attack detection in advanced metering infrastructure, IEEE Trans. Instrum. Meas. 73 (2024) 1–12.

[112] P. Gulganwa, S. Jain, Base station model selection using machine learning technique for wireless sensor network, Wirel. Pers. Commun. 132 (2) (2023) 1225–1239.

[113] P. Sen, S. Waghmare, Machine learning based intrusion detection system for Real-Time smart grid security, in: 2021 13th IEEE PES Asia Pacific Power & Energy Engineering Conference, APPEEC, 2021, pp. 1–6.

[114] T. Wang, N. Huang, Y. Wu, J. Gao, T.Q.S. Quek, Latency-Oriented secure wireless federated learning: A channel-sharing approach with artificial jamming, IEEE Internet Things J. 10 (11) (2023) 9675–9689.

[115] S. Smadi, N. Aslam, L. Zhang, Detection of online phishing email using dynamic evolving neural network based on reinforcement learning, Decis. Support Syst. 107 (2018) 88–102.

[116] L. Yang, A. Shami, A lightweight concept drift detection and adaptation framework for IoT Data Streams, IEEE Internet Things Mag. 4 (2) (2021) 96–101.

[117] C. Fu, Q. Li, M. Shen, K. Xu, Realtime robust malicious traffic detection via frequency domain analysis, in: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, CCS '21, Association for Computing Machinery, New York, NY, USA, 2021, pp. 3431–3446.

[118] R. Mohammadi, C. Lal, M. Conti, HTTPScout: A machine learning based countermeasure for HTTP flood attacks in SDN, Int. J. Inf. Secur. (2022).

[119] S. Zhou, M. Du, X. Liu, H. Shen, Algorithm for community security risk assessment and influencing factors analysis by back propagation neural network, Heliyon 10 (9) (2024) e30185.

[120] V. Badarla, C. Siva Ram Murthy, A novel learning based solution for efficient data transport in heterogeneous wireless networks, Wirel. Netw. 16 (6) (2010) 1777–1798.

[121] K. Bian, R. Priyadarshi, Machine learning optimization techniques: A survey, classification, challenges, and future research issues, Arch. Comput. Methods Eng. (2024) ISSN: 1886-1784.

[122] A. Mozo, A. Karamchandani, L. de la Cal, S. Gómez-Canaval, A. Pastor, L. Gifre, A Machine-Learning-Based cyberattack detector for a Cloud-Based SDN controller, Appl. Sci. 13 (8) (2023).

[123] A. Paya, D.C. Marinescu, Energy-Aware load balancing and application scaling for the cloud ecosystem, IEEE Trans. Cloud Comput. 5 (1) (2017) 15–27.

[124] P. Gulganwa, S. Jain, EE-WCA: Energy efficient weighted clustering algorithm to regulate application's quality of service requirements, Wirel. Pers. Commun. 124 (4) (2022) 3647–3660.
[125] J. Ko, Y.-J. Choi, R. Paul, Computation offloading technique for energy efficiency of smart devices, J. Cloud Comput. 10 (1) (2021) 44.
[126] E. Le Sueur, G. Heiser, Dynamic voltage and frequency scaling: the laws of diminishing returns, in: Proceedings of the 2010 International Conference on Power Aware Computing and Systems, HotPower '10, USENIX Association, USA, 2010, pp. 1–8.
[127] F. Jameel, S. Wyne, G. Kaddoum, T.Q. Duong, A comprehensive survey on cooperative relaying and jamming strategies for Physical Layer Security, IEEE Commun. Surv. Tutor. 21 (3) (2019) 2734–2771.
[128] A. Majeed, N.B. Abu-Ghazaleh, Packet aggregation in multi-rate wireless LANs, in: 2012 9th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, SECON, 2012, pp. 452–460.
[129] S.L. Zou, S.B. Goyal, A.S. Rajawat, G. Paramasivam, Advancing network security: A big data analytics approach to trust and integrity, in: A. Swaroop, V. Kansal, G. Fortino, A.E. Hassanien (Eds.), Proceedings of Fifth Doctoral Symposium on Computational Intelligence, Springer Nature Singapore, Singapore, 2024, pp. 481–492.
[130] A.L. Blum, P. Langley, Selection of relevant features and examples in machine learning, Artificial Intelligence 97 (1) (1997) 245–271, Relevance.
[131] A. Benkessirat, N. Benblidia, Fundamentals of feature selection: An overview and comparison, in: 2019 IEEE/ACS 16th International Conference on Computer Systems and Applications, AICCSA, 2019, pp. 1–6.
[132] I. Gridin, Model pruning, in: Automated Deep Learning using Neural Network Intelligence: Develop and Design PyTorch and TensorFlow Models using Python, A Press, Berkeley, CA, 2022, pp. 319–355.
[133] Y. Huang, Y. Cheng, A. Bapna, O. Firat, M.X. Chen, D. Chen, H. Lee, J. Ngiam, Q.V. Le, Y. Wu, Z. Chen, Gpipe: efficient training of giant neural networks using pipeline parallelism, in: Proceedings of the 33rd International Conference on Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, USA, 2019.
[134] J. Wu, C. Leng, Y. Wang, Q. Hu, J. Cheng, Quantized convolutional neural networks for mobile devices, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2016, pp. 4820–4828.
[135] E. Wang, J.J. Davis, D. Moro, P. Zielinski, J.J. Lim, C. Coelho, S. Chatterjee, P.Y.K. Cheung, G.A. Constantinides, Enabling binary neural network training on the edge, Vol. 22 (6), (2023).
[136] G. Abdelmoumin, D.B. Rawat, A. Rahman, On the performance of machine learning models for Anomaly-Based intelligent intrusion detection systems for the Internet of Things, IEEE Internet Things J. 9 (6) (2022) 4280–4290.
[137] P. Duboue, Feature engineering, in: R. Egger (Ed.), Applied Data Science in Tourism: Interdisciplinary Approaches, Methodologies, and Applications, Springer International Publishing, Cham, 2022, pp. 109–127.
[138] G.P. Dubey, D.R.K. Bhujade, Optimal feature selection for machine learning based intrusion detection system by exploiting attribute dependence, Mater. Today: Proc. 47 (2021) 6325–6331, SI: TIME-2021.
[139] C. Cheng, W.P. Tay, G.-B. Huang, Extreme learning machines for intrusion detection, in: The 2012 International Joint Conference on Neural Networks, IJCNN, 2012, pp. 1–8.
[140] A.O. Salau, S. Jain, Feature extraction: A survey of the types, techniques, applications, in: 2019 International Conference on Signal Processing and Communication, ICSC, 2019, pp. 158–164.
[141] J. Pearl, Heuristics: Intelligent search strategies for computer problem solving, 1984.
[142] H. Karau, A. Konwinski, P. Wendell, M. Zaharia, Learning Spark: Lightning-Fast Big Data Analytics, first ed., O'Reilly Media, Inc., 2015.
[143] M.O. Mendonça, S.L. Netto, P.S. Diniz, S. Theodoridis, Chapter 13 - machine learning: Review and trends, in: P.S. Diniz (Ed.), Signal Processing and Machine Learning Theory, Academic Press, 2024, pp. 869–959.
[144] D. Vasan, E.J.S. Alqahtani, M. Hammoudeh, A.F. Ahmed, An AutoML-based security defender for industrial control systems, Int. J. Crit. Infrastruct. Prot. 47 (2024) 100718.
[145] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, B.B. Gupta, X. Chen, X. Wang, A survey of deep active learning, ACM Comput. Surv. 54 (9) (2021).
[146] A. Shahraki, M. Abbasi, A. Taherkordi, A.D. Jurcut, Active learning for network traffic classification: A technical study, IEEE Trans. Cogn. Commun. Netw. 8 (1) (2022) 422–439.
[147] X. He, K. Zhao, X. Chu, AutoML: A survey of the state-of-the-art, Knowl.-Based Syst. 212 (2021) 106622.
[148] M. Kim, W. Jang, J. Hur, M. Yoon, Relative frequency-rank encoding for unsupervised network anomaly detection, IEEE/ACM Trans. Netw. 32 (4) (2024) 3453–3467.
[149] S. Marchal, M. Miettinen, T.D. Nguyen, A.-R. Sadeghi, N. Asokan, AuDI: Toward autonomous IoT Device-Type identification using periodic communication, IEEE J. Sel. Areas Commun. 37 (6) (2019) 1402–1412.
[150] C. Slimani, L. Morge-Rollet, L. Lemarchand, F. Le Roy, D. Espes, J. Boukhobza, Characterizing intrusion detection systems on heterogeneous embedded platforms, in: 2023 26th Euromicro Conference on Digital System Design, DSD, 2023, pp. 278–285.
[151] A. Valero León, INsIDES: A new machine learning-based intrusion detection system, 2017, http://hdl.handle.net/10230/32875.
[152] G. Frishman, Y. Ben-Itzhak, O. Margalit, Cluster-Based load balancing for better network security, in: Proceedings of the Workshop on Big Data Analytics and Machine Learning for Data Communication Networks, in: Big-DAMA '17, Association for Computing Machinery, New York, NY, USA, 2017, pp. 7–12.
[153] X. Tan, T. Li, R. Chen, F. Liu, L. Zhang, Challenges of using Pre-trained models: the practitioners' perspective, in: 2024 IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER, 2024, pp. 67–78.
[154] Getting started with streamline: Introduction to streamline, 2024, https://developer.arm.com/documentation/101816/0707/Getting-started-with-Streamline/Introduction-to-Streamline. (Accessed 04 January 2024).
[155] POWMON: Power monitoring technologies, 2024, https://www.arm.ecs.soton.ac.uk/technologies/powmon/. (Accessed 04 January 2024).
[156] S. Li, J.H. Ahn, R.D. Strong, J.B. Brockman, D.M. Tullsen, N.P. Jouppi, McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures, in: 2009 42nd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO, 2009, pp. 469–480.
[157] V.M. Weaver, M. Johnson, K. Kasichayanula, J. Ralph, P. Luszczek, D. Terpstra, S. Moore, Measuring energy and power with PAPI, in: 2012 41st International Conference on Parallel Processing Workshops, 2012, pp. 262–268.
[158] ResMon - computer hope, 2024, https://www.computerhope.com/jargon/r/resmon.html. (Accessed 04 January 2024).
[159] Fil: A memory profiler for Python, 2024, https://pythonspeed.com/fil/docs/index.html. (Accessed 19 January 2024).
[160] Sciagraph: Identify speed and memory bottlenecks in Python data processing workflows, 2024, https://www.sciagraph.com/. (Accessed 19 January 2024).
[161] Time — Time access and conversions, 2024, https://docs.python.org/3/library/time.html. (Accessed 19 January 2024).
[162] R. Patgiri, Issues and challenges in Big Data: A survey, in: A. Negi, R. Bhatnagar, L. Parida (Eds.), Distributed Computing and Internet Technology, Springer International Publishing, Cham, 2018, pp. 295–300.
[163] R. Vinayakumar, K.P. Soman, P. Poornachandran, Applying convolutional neural network for network intrusion detection, in: 2017 International Conference on Advances in Computing, Communications and Informatics, ICACCI, 2017, pp. 1222–1228.
[164] J. Gama, I. Žliobaitundefined, A. Bifet, M. Pechenizkiy, A. Bouchachia, A survey on concept drift adaptation, ACM Comput. Surv. 46 (4) (2014).
[165] A. Mahadevan, M. Mathioudakis, Cost-aware retraining for machine learning, Knowl.-Based Syst. 293 (2024) 111610.
[166] R. Joshi, R.S. Somesula, S. Katkoori, Empowering Resource-Constrained IoT edge devices: A hybrid approach for edge data analysis, in: D. Puthal, S. Mohanty, B.-Y. Choi (Eds.), Internet of Things. Advances in Information and Communication Technology, Springer Nature Switzerland, Cham, 2024, pp. 168–181.
[167] Y. Abadade, A. Temouden, H. Bamoumen, N. Benamar, Y. Chtouki, A.S. Hafid, A comprehensive survey on TinyML, IEEE Access (2023).
[168] P. Shah, Y. Govindarajulu, P. Kulkarni, M. Parmar, Enhancing TinyML security: Study of adversarial attack transferability, 2024, arXiv preprint arXiv:2407.11599.
[169] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, S. Lloyd, Quantum machine learning, Nat. 549 (7671) (2017) 195–202.
[170] M. Schuld, I. Sinayskiy, F. Petruccione, An introduction to quantum machine learning, Contemp. Phys. 56 (2) (2015) 172–185.
[171] M. Cerezo, G. Verdon, H.-Y. Huang, L. Cincio, P.J. Coles, Challenges and opportunities in quantum machine learning, Nat. Comput. Sci. 2 (9) (2022) 567–576.
[172] U.C. Çabuk, M. Tosun, R.H. Jacobsen, O. Dagdeviren, A holistic energy model for drones, in: 2020 28th Signal Processing and Communications Applications Conference, SIU, 2020, pp. 1–4.
[173] N. Pitropakis, E. Panaousis, T. Giannetsos, E. Anastasiadis, G. Loukas, A taxonomy and survey of attacks against machine learning, Comput. Sci. Rev. 34 (2019) 100199.
[174] J. Suomalainen, A. Juhola, S. Shahabuddin, A. Mämmelä, I. Ahmad, Machine learning threatens 5G security, IEEE Access 8 (2020) 190822–190842.
[175] Y. Sun, J. Liu, J. Wang, Y. Cao, N. Kato, When machine learning meets privacy in 6G: A survey, IEEE Commun. Surv. Tutor. 22 (4) (2020) 2694–2724.
[176] G. Sriramanan, S. Addepalli, A. Baburaj, et al., Towards efficient and effective adversarial training, Adv. Neural Inf. Process. Syst. 34 (2021) 11821–11833.
[177] O. Sagi, L. Rokach, Ensemble learning: A survey, Wiley Interdiscip. Rev.: Data Min. Knowl. Discov. 8 (4) (2018) e1249.
[178] L. Weber, S. Lapuschkin, A. Binder, W. Samek, Beyond explaining: Opportunities and challenges of XAI-based model improvement, Inf. Fusion 92 (2023) 154–176.
[179] J. Moysen, L. Giupponi, From 4G to 5G: Self-organized network management meets machine learning, Comput. Commun. 129 (2018) 248–268.
[180] V. Kulothungan, Securing the AI Frontier: Urgent ethical and regulatory imperatives for AI-Driven cybersecurity, in: 2024 IEEE International Conference on Big Data (BigData), 2024, pp. 5602–5609.

Md Muzammal Hoque is a research scientist at the VTT Technical Research Centre of Finland. He received his M.Sc. in Information and Communication Technology (Cybersecurity) from the University of Turku, Finland, in 2024. He also obtained an M.Sc. (Engineering) and a B.Sc. (Engineering) in Information and Communication Engineering from the University of Rajshahi, Bangladesh, in 2017 and 2018, respectively. He received the University of Turku Scholarship in 2021 and the National Science and Technology (NST) fellowship from the Ministry of Science and Technology (MOST), Bangladesh, in 2019. His research interests include cybersecurity, machine learning, 5G and 6G security, and adaptive and sustainable cybersecurity technologies.

Ijaz Ahmad is a senior scientist at the VTT Technical Research Centre of Finland, and Adjunct Professor (Docent) at the University of Oulu, Finland. He received his MSc. and PhD in Wireless Communications from the University of Oulu, Finland, in 2012 and 2018, respectively. He is the recipient of over 13 awards (Finnish and international) for excellence in research including the Nokia foundation, Tauno Tönning and Jorma Ollila grant awards, and two IEEE best paper awards. His research interests include 5G and 6G security, security of critical environments, the applications of machine learning in network security, and sustainability of cybersecurity technologies.

Jani Suomalainen is a senior scientist at the VTT Technical Research Centre of Finland in Espoo. He received his M.Sc. degree from Lappeenranta University of Technology and his D.Sc. degree from Aalto University, Finland. He is specialized in cyber and network security and has over twenty years of experience on different cybersecurity topics. Recently, he has participated to several European and Finnish cooperation projects studying security in 5G/6G networks and cybersecurity applications of machine learning. His research interests include threat modelling, security architectures, and intelligent security solutions.

Paolo Dini is Leading Researcher at the Centre Tecnologic de Telecomunicacions de Catalunya (CTTC) and coordinates the activities of the Sustainable AI research unit. He received M.Sc. and Ph.D. from the University of Roma La Sapienza. His current research interests include machine learning, multi-agent systems, sustainable computing for cyber–physical systems, distributed optimization and optimal control, energy efficiency. He has been involved in more than 25 research and development projects on these topics, coordinating many of them. The MSCA SCAVENGE (coordinator) and MSCA GREENEDGE (research officer) have been included in the EU Success Stories. He is member of the ELLIS Society.

Mohammad Tahir (Senior Member, IEEE) is a University Lecturer and Adjunct professor in Cybersecurity Engineering at the University of Turku. He received the M.Sc. and Ph.D. degrees in electrical and computer engineering from the Department of Electrical and Computer Engineering, International Islamic University Malaysia, in 2011 and 2016, respectively. Before transitioning to academia, Dr. Tahir accumulated seven years of industry experience with R&D organizations, where he contributed to projects involving software-defined radios for efficient spectrum utilization, development of IoT-based solutions and co-authoring patents in wireless networks and IoT. His current research focuses on 5G/6G, IoT, cybersecurity and machine learning applications in wireless networks and cybersecurity.