



# *MIDES: A Multi-layer Intrusion Detection System using Ensemble Machine Learning*

Article

Accepted version

V. Agate, A. De Paola, P. Ferraro, and G. Lo Re

International Journal of Intelligent Networks
DOI: 10.1016/j.ijin.2025.09.001



Publisher: KeAi

# MIDES: A Multi-layer Intrusion Detection System using Ensemble Machine Learning

Vincenzo Agate[a,b], Alessandra De Paola[a,b], Pierluca Ferraro[a,b], Giuseppe Lo Re[a,b]

[a] *Department of Engineering, University of Palermo*
[b] *Cybersecurity National Lab CINI - Consorzio Interuniversitario Nazionale per l'Informatica*

**Abstract**

In recent years, as the frequency and types of network attacks increase, Intrusion Detection Systems (IDSs) have become essential components of most organizations' security infrastructure. Although the use of machine learning methods shows great promise for the design of effective IDSs, existing methods still have several limitations. Single classifiers are never able to recognize all types of attacks, regardless of the underlying algorithm. This paper proposes MIDES, a novel multi-layer IDS that integrates binary, multi-class, and meta-classifiers into a flexible architecture. MIDES employs a fast binary classifier to filter clearly benign traffic, an ensemble of specialized multi-class classifiers to analyze suspicious events, and a meta-classification layer to refine decisions. A self-adaptive agent dynamically selects the most appropriate decision strategy for each input using both static and dynamic heuristics. The system is designed to be extensible, adaptable to evolving threats, and efficient in real-time scenarios. The proposed system has been extensively evaluated on the well-known CIC-IDS2017 and CSE-CIC-IDS2018 public datasets and compared against state-of-the-art works, showing that MIDES achieves high accuracy across all 14 attack classes while significantly reducing classification time, outperforming the compared systems.

*Keywords:* Intrusion Detection Systems, Ensemble Machine Learning, Stacked Generalization, Meta-Learning

## 1. Introduction

Today, due to the increasing adoption of ICT technologies, it is common to face cyber attacks that put the critical infrastructure of various organisations at risk. A successful attack can lead to serious consequences, such as financial losses, service interruptions and disclosure of confidential information. One of the most advanced solutions to mitigate such dangerous activities is the use of Intrusion Detection Systems (IDS), whose objective is to independently identify and prevent attacks and misuse of networks by analysing information from heterogeneous sources, such as network traffic patterns, user activity patterns, and data extracted from system logs.

However, existing IDS solutions often rely on fixed decision mechanisms or static ensemble strategies that fail to adapt to varying attack behaviors and evolving data distributions. These limitations highlight the need for a more flexible and self-adaptive detection framework. In this work, we address this gap through a novel architecture that can dynamically reconfigure its detection pipeline at runtime.

In order to effectively protect the network from intruders attempting to gain unauthorized access to system resources, IDSs must be able to detect all attack attempts, which corresponds to minimising the number of false negatives, i.e. the number of intrusions mistakenly recognised as legitimate behaviour, since even a single well-executed attack can lead to fatal consequences. However, at the same time, in order to ensure adequate usability of the system, it is necessary to minimise the number of false positives, i.e. the number of legitimate actions wrongly considered as intrusions [57]. Indeed, too many false positives could lead administrators to underestimate the frequent false alarms and do not react properly in the event of a real attack.

With respect to these objectives, the most effective solution is the adoption of anomaly-based IDSs, which are trained to recognise normal user behaviour and to consider any significant deviation as an anomaly and thus potentially an intrusion [55]. To design anomaly-based IDSs, several works in the literature propose systems based on machine learning techniques that claim good detection performance. Nevertheless, an accurate analysis of the experimental results highlights a critical issue, well known in the literature, that affects common machine learning methods. Namely, although the overall performance of a classifier is fairly good, its specific performance in detecting each individual attack class can vary greatly. This problem is evident in standard benchmark datasets, where classifiers often perform very well on dominant attack classes but poorly on rare or stealthy ones such as infiltration [58]. Therefore, extremely good *global* predictive performances may not be representative of the real performance of a

*Email addresses:* vincenzo.agate@unipa.it (Vincenzo Agate), alessandra.depaola@unipa.it (Alessandra De Paola), pierluca.ferraro@unipa.it (Pierluca Ferraro), giuseppe.lore@unipa.it (Giuseppe Lo Re)

classifier. Simply put, there is no single universally valid machine learning approach that can adapt to all contexts. This is largely caused by the extreme variability of attacks and network traffic. The result is that classifiers based on known machine learning techniques exhibit widely different levels of performance for different attack classes.

A promising approach to solve such a problem is the adoption of an ensemble of classifiers with different behaviours and characteristics in order to achieve better results than a single classifier. However, such an approach could increase computational load, potentially affecting system efficiency. Moreover, static ensembles rely on fixed combinations of classifiers and decision rules, which may be optimal only if the analyzed data have the same distribution seen during training. These fixed combinations can lead to significant performance degradation, especially in detecting rare or evolving threats [61]. In fact, one of the requirements of IDSs is the ability to perform real-time analysis, which is necessary to efficiently analyse the traffic flow the system receives, in order to promptly identify incoming threats and immediately apply appropriate countermeasures.

Moreover, the solutions proposed in the literature, which adopt an ensemble learning approach, select the set of classifiers to be adopted through a trial-and-error process, and lack a formalised methodology that can drive the design process even in different scenarios. In contrast to previous methods, the proposed system is designed to dynamically select its decision strategy based on input characteristics and performance considerations. To overcome the limitations of static classifier ensembles, this work proposes MIDES, a multi-layer Intrusion Detection System that exploits ensemble machine learning through a combination of binary, multi-class, and meta-classifiers, coordinated via an adaptive agent. This architecture departs from fixed ensemble designs by dynamically adjusting the decision flow at runtime, according to performance and confidence metrics.

MIDES is designed to overcome the limitations of single classifiers and traditional ensemble methods in detecting specific attacks while keeping training and inference times low. As shown in the experimental section, the combination of multiple classifiers effectively improves detection performance over individual state-of-the-art methods. The selection of specific classifiers representing the building blocks of MIDES was carried out through a well-defined methodology that can be adopted to design specific IDSs in very different scenarios.

The novel contributions of this work are manyfold:

- The architecture of MIDES, a novel multi-layer Intrusion Detection System that is based on the combination of multiple classifiers, and is designed to minimize the computational load of the system.
- The adoption of a combination of binary, multi-class, and meta-classifiers to improve the classification performance of the IDS.
- A well-defined methodology adopted to select the specific building blocks of MIDES, enabling its adaptability to different scenarios.
- A set of novel heuristics based on static and dynamic components which allow MIDES to reason about itself and its own performance and choose the best combination of classifiers to use for each input event.

The experimental evaluation extensively tested MIDES on two widely adopted datasets with a series of experiments that validated all components of the system. Furthermore, MIDES was compared to other state-of-the-art works, outperforming them in both accuracy and classification time.

The remainder of this paper is organized as follows. Section 2 outlines other relevant work in literature. Section 3 describes MIDES in detail, providing a thorough discussion of its main components. Section 4 presents the experimental evaluation and comparison with state-of-the-art work, which allowed for a thorough testing of all system components. Finally, Section 5 reports some conclusions and outlines some possible future developments.

## 2. Related Work

The purpose of IDSs is to maintain adequate protection in a computer system and detect adverse events to ensure that any anomalous network traffic that escapes traditional firewalls can be stopped immediately. Despite the wide range of IDSs proposed in the literature, one of the main open challenges is building systems that can generalize across multiple attack classes, adapt to evolving threats, and maintain low false positives, especially in multi-class and real-world scenarios with imbalanced datasets. In recent years, the rapid evolution of attack techniques and the growing diversity of network environments have exposed the limitations of traditional IDS approaches. In particular, many recent IDSs fail to support real-time adaptation, dynamic ensemble selection, or robust multi-class discrimination in realistic and imbalanced settings. This motivates the need for architectures that combine high accuracy with flexibility and computational efficiency, particularly in heterogeneous or rapidly changing scenarios.

*Background*

Two major categories of IDS have been extensively studied by researchers over the years, namely Signature-based Intrusion Detection System and Anomaly-based Intrusion Detection System [33, 15, 36]. Signature-based IDSs exploit signature matching techniques to detect intrusions that have already occurred. In other words, since every intrusion leaves a signature in the violated system, the detection problem becomes a recognition problem over a database of known attack signatures. Recent works dealing with signature-based IDS are proposed in [51, 30, 44].

The major advantage of signature-based IDSs is that they are able to achieve excellent accuracy in the presence of a known attack whose signature has already been identified. They are also known to exhibit a very low false positive rate when recognizing attacks, which is important for IDSs because an excessive number of false positives would cause a decrease in the level of trust in the system.

Unfortunately, signature-based IDSs show poor performance in the presence of modern and more sophisticated attacks where there is no signature available, for example in the case of zero-day attacks [31]. Zero-day threats can potentially be solved by anomaly-based IDSs, which differentiate network events based on the type of behavior deemed acceptable or anomalous.

For an anomaly-based IDS, it is of paramount importance that the system learns to recognize behaviors that exhibit a significant deviation from normal such that they can be considered anomalous [24, 52]. Over the years, different techniques have been adopted trying to achieve this capability [25, 27, 87, 22, 14, 71, 70]. The main categories of anomaly-based IDS are based on different approaches, such as statistical techniques, knowledge-based techniques, and machine learning-based techniques.

Statistical techniques allow an IDS to build a probability distribution of a normal behavior profile and identify low probability events as potentially malicious [32, 76]. Metrics used to quantify behavior that is unlikely or far from the normal profile usually include mean, median, and standard deviation calculated on packets or features extracted from network traffic. Several works exploit time-series analyses to determine patterns of normal use and eventually report system misuse when observations are found to be unlikely in the time interval under consideration [29]. The main problems with statistics-based techniques concern the difficulty in estimating distributions on high-dimensional data, which is typical of network traffic.

Knowledge-based techniques, on the other hand, leverage a knowledge base built from traffic by reconstructing legitimate behavioral profiles. Anything that differs from the legitimate profile is treated as an intrusion. Unlike other techniques proposed in the literature, the profile constructed depends on a set of rules based on human knowledge that attempt to describe legitimate behavior. Some work has exploited Finite-State Machines to capture and model legitimate behavior profiles [73, 1]. The main weakness of this type of technique is the difficulty in defining rules and profiles that are capable of detecting modern attacks, which are increasingly sophisticated and constantly evolving [33].

The third and most promising category of anomaly-based IDSs is based on Machine Learning techniques and employs both complex and simple learning models to perform automatic knowledge extraction from network traffic data to limit the effort of human operators in profiling the legitimate use of network resources [45]. As in other research areas, two major categories of ML have been widely employed, namely supervised and unsupervised learning techniques. Unsupervised learning techniques exploit the use of an unlabeled data set, whereby data are often automatically grouped according to a criterion (e.g., exploiting distance metrics). One of the most explored techniques of unsupervised learning in intrusion detection is the K-means [48, 47, 13], which separates the elements of a dataset into K clusters by grouping them based on some similarity criterion, such as the Euclidean distance. Events sufficiently distant from the cluster centroids are considered intrusions.

Supervised learning techniques exploit datasets containing labeled network traffic (e.g., benign, malicious, and other, or specifying the various classes of attack). Since very often the network datasets used have a large number of features, dimensionality reduction is achieved through feature selection techniques [81] such as principal component analysis (PCA) [8, 60] and information gain (IG) [21]. Supervised ML techniques used in IDS include decision trees (DT) [23, 12], Naive Bayes [34, 49], genetic algorithms [50, 11], artificial neural networks (ANN) [17, 6], support vector machines (SVM) [35], Hidden Markov Model (HMM) [54] and K-Nearest Neighbors (KNN) [37].

Each of these techniques has drawbacks that do not allow them to be used as the only learning model that can correctly recognize all types of intrusions. Currently, the most promising direction to achieve overall good performance seems to be the use of ensemble learning techniques [2], which exploit multiple machine learning algorithms to achieve better performance than those achieved individually. In the following, some recent work based on ensemble techniques are reported. In the following, we categorize recent IDS approaches into three main groups: static ensemble-based systems, adaptive ensemble and meta-learning IDSs, IoT and domain-specific solutions. This organization contributes to highlighting the novelty of MIDES, which combines the robustness of ensembles with dynamic decision-making and stacked generalization.

*Static ensemble-based IDSs*

The authors of [67] propose an anomaly-based IDS that relies on a two-stage meta classifier, i.e. a rotation forest and bagging; in order to take into account only relevant features, they adopted a hybrid feature selection method based on particle swarm optimization, ant colony algorithm, and genetic algorithm. Despite its good overall performance, however, the proposed system is unable to distinguish between different attack classes. In [90], in order to choose the best combination of features and reduce the dimensionality of the data, a heuristic algorithm called CFS-BA has been proposed. The authors introduce an ensemble approach that combines C4.5, Random Forest (RF), and Forest by Penalizing Attributes (Forest PA) algorithms. Finally, a voting technique is used to combine the probability distributions of the base classifiers for attack recognition. Although the system achieves good performance, the authors themselves state that it could be further improved to deal with rare attacks.

Table 1: Comparison of major recent anomaly-based IDSs proposed in the literature that exploit ensemble learning.

| Systems | ML techniques | Multi-class | # of attack classes | Datasets |
|---|---|---|---|---|
| Tama et al. [67] | Rotation forest and Bagging | - | - | NSL-KDD, UNSW-NB15 |
| Zhou et al. [90] | C4.5, RF, and ForestPA | ✓ | 5 | NSL-KDD, CIC-IDS2017, AWID |
| Alhowaide et al. [9] | SVM, SGD, Bernoulli-NB, Gaussian-NB, DT, KNN, Logistic Regr., and MLP | - | - | NSL-KDD, UNSW-NB15, BoTNeTIoT, BoTIoT |
| Seth et al. [64] | KNN, DT, RF, Extra Trees, XGBoost, Hist. Based Gradient Boosting, LightGBM | ✓ | 6 | CSE-CIC-IDS2018 |
| Li et al. [38] | DT, KNN, Naive Bayes, and SVM | ✓ | 4 | NSL-KDD, Key Infrastructure Protection Center |
| Saiyed and Al-Anbagi [62] | CNN, LSTM, Pruning | - | - | HL-IoT, ToN-IoT, CICIDS-17, and ISCX-12 |
| Sayem et al. [63] | LSTM, GRU, CNN, DNN | ✓ | 6 | UNSW-15, CICIDS-2017 |
| Lin et al. [39] | BLS, OCBLS | - | - | NSL-KDD, UNSW-NB15 |
| Yang et al. [84] | XGBoost, LightGBM, CatBoost | ✓ | 6 | CIC-IDS2017, Car-Hacking |
| Yu et al. [86] | Ensemble of Transformer Encoders | ✓ | 5 | NSL-KDD, CICIDS2017 |
| Yang et al. [85] | Variational AutoEncoders and Knowledge Distillation | ✓ | 6 | Gas Pipeline, Water Storage Tank, CICIDS2017, NSL-KDD |
| MIDES | DT, RF, Naive-Bayes, NN, adaptive meta-classifier | ✓ | 14 | CIC-IDS2017, CSE-CIC-IDS2018 |

An ensemble framework that can detect different attack categories was recently proposed in [64]. The solution ranks the detection ability of different base classifiers according to the $F_1$-score obtained for different attack categories, to identify various types of attacks. The output for the final prediction algorithm is only considered if the classifier has the highest $F_1$-score for that particular attack category. However, the results show that they achieve good classification performance with 6 different attack classes out of 14 in the dataset.

*Adaptive ensemble and meta-learning IDSs*

Adaptive ensemble and meta-learning techniques have been increasingly adopted in the design of modern IDSs to enhance detection performance and flexibility across diverse attack scenarios [88]. The authors of [38] propose a model based on sustainable ensemble learning and on incremental learning. Such a system exploits multiclass regression models so that the ensemble is adapted to recognize different types of attacks and by means of an iterative update method, the parameters and decision results of the historical model are added to the training process of the final ensemble model. ENIDS, a framework proposed in [63], leverages several techniques such as CNN, LSTM and Gated Recurrent Units (GRUs) in a stacked ensemble model, where the base learners are trained independently and their pre-trained weights are used to train a DNN meta-learner. While this approach aims to reduce computational overload and improve detection performance for various types of cyberattacks in network traffic, it struggles to distinguish new types of traffic from legitimate traffic or other intrusions if such traffic was not present in the training data. MC-OCBLS, proposed in [39], introduces a novel hybrid ensemble framework that combines broad learning systems with multiple classifiers to enhance the accuracy and robustness of intrusion detection. Moreover, the model employs the maximum correntropy criterion to improve the ability to handle non-Gaussian noise. Despite its promising results, the model faces limitations in incremental learning, dimensionality reduction, and multiclass anomaly detection, highlighting areas for future research and improvement.

Recently, Transformer-based architectures have gained increasing attention in the intrusion detection field due to their strong ability to capture long-range dependencies and contextual information in network traffic data, achieving competitive results especially in detecting complex and stealthy attacks [80, 26].

NAEF (Nonlinear Augmented Explicit Features) [86] addresses cross-domain intrusion detection in low-data scenarios by expanding the feature space explicitly and modeling nonlinear mappings across domains. It leverages a Transformer-based ensemble classifier with attention mechanisms for feature selection and data balancing. However, its evaluation is limited to five categories (including DoS, DDoS, Probe, and Password attacks), ex-

cluding other attack types and raising concerns about the method's generalizability to more diverse intrusion scenarios.

However, existing adaptive ensemble and meta-learning IDSs typically require frequent retraining. Moreover, many solutions are evaluated on a reduced number of attack classes and lack decision-level adaptability at runtime, making them less suitable for large-scale, multi-class scenarios with imbalanced traffic.

*IoT and domain-specific IDS solutions*

An ensemble classification model designed for the IoT scenario using an automatic Model Selection Method (MSM) is proposed in [9]. While the authors provide significant advances in IoT IDSs, they considered cyber-threats as a binary classification problem and tested the proposed methods and models only on session-based datasets.

The work proposed in [62] presents a novel approach to attack detection in IoT networks using a system named DEEPShield. This system integrates a Convolutional Neural Network (CNN) and a Long Short-Term Memory (LSTM) network with a traffic analysis system to improve the accuracy of high and low volume attack detection in resource-constrained environments. However, the system has only been tested in the detection of DDoS attacks.

To accurately detect various types of attacks in Internet of Vehicles (IoV) networks, the authors of [84] propose an ensemble IDS framework called Leader Class and Confidence Decision Ensemble (LCCDE). This system is designed to select the best performing ML model among three advanced ML algorithms (XGBoost, LightGBM, and CatBoost) for each class or type of attack. The class-leading models with their prediction confidence scores are then used to make accurate decisions regarding the detection of various types of cyber-attacks. Given its excellent performance, it is used as a comparison for MIDES. However, as will be shown in the experimental section, LCCDE's leading model selection adds considerable overhead, slowing down the overall execution time.

The authors of [85] propose a lightweight and dynamic open-set intrusion detection framework for IIoT that combines Variational AutoEncoders (VAE) and Extreme Value Theory (EVT) to identify known and unknown attacks, and employs knowledge distillation (KD) for efficient model updates. The method enables continual learning while minimizing computational and memory overhead. However, its performance is sensitive to the degree of similarity between unknown and known attacks, and detection can degrade when reconstruction errors of unknown samples overlap with those of known classes, potentially limiting effectiveness in complex, real-world scenarios with subtle or evolving attack patterns.

AILL-IDS [28] is a recent VANET-specific IDS that employs incremental semi-supervised learning to reduce labeling effort while maintaining high detection performance. Unlike our approach, which adapts at the decision level through an internal arbitration mechanism, AILL-IDS relies on periodic model updates based on streaming data.

Although recent studies have proposed innovative IDS solutions tailored to specific application domains such as vehicular networks [74, 77, 68], cross-domain communications [86], consumer electronics [10, 56], smart grids [42], adaptive network monitoring [89, 40], industrial control systems [82], and secure healthcare networks [3], these approaches often rely on domain-specific assumptions and features, which may limit their applicability to broader or dynamic network environments.

*Comparative analysis of dynamic and meta-learning IDS frameworks*

While ensemble methods generally improve detection accuracy, many IDSs still exhibit limited multi-class capabilities or operate under simplified assumptions, such as binary classification or balanced datasets [5, 9, 4, 64]. Only a few frameworks integrate adaptivity or meta-learning in a way that supports efficient and flexible decision-making. For example, the class-wise ranking strategy in Seth et al. [64] improves detection granularity but is evaluated on a limited number of attack categories and lacks mechanisms for timely adaptation to novel threats. Similarly, LCCDE [84] achieves competitive per-class results, yet introduces significant computational overhead and does not support real-time decision reconfiguration. Other systems adopt a more structured meta-learning approach. ENIDS [63], for instance, uses a stacked ensemble with a deep meta-learner trained on base models, but operates over a fixed ensemble and requires re-training to adapt to new patterns. Incremental-learning models such as that proposed by Li et al. [38] address long-term adaptability via model updates, but are still constrained by their reliance on periodic updates and lack support for decision-level flexibility during inference. In contrast, MIDES combines the robustness of ensemble learning with a lightweight and modular arbitration mechanism that operates at runtime. Without incurring the costs of class-wise ranking [64, 84] or full re-training [38], MIDES dynamically selects between multiple decision modules based on real-time confidence and expertise scoring. This design enables effective multi-class classification across 14 attack categories and provides increased flexibility in responding to changing network conditions.

Table 1 summarizes a comparative analysis of the discussed systems. It highlights how most of the recent works are limited to detecting a reduced set of attack classes, or rely on rigid ensemble structures that lack runtime adaptivity. In contrast, MIDES addresses these limitations by supporting multi-class classification across 14 attack types, while incorporating a formal methodology for adaptive classifier selection and execution. This positions MIDES as a comprehensive and extensible solution capable of maintaining high accuracy and responsiveness in complex and evolving threat landscapes.

## 3. MIDES Architecture

Although most machine learning algorithms used to design IDSs provide high performance in terms of accuracy in classifying specific attack classes, most of those that leverage a single classifier often struggle against other attack classes. This problem is faced by the "No Free Lunch Theorem" (NFL) [79], which states that it is impossible to find a general-purpose function $f$ that can provide a universal optimization strategy, thus necessitating, for each application domain, the study of a function $f^*$ that specializes in solving the specific problem.

The approach proposed here is the adoption of meta-learning, i.e. exploiting meta-knowledge about machine learning models to more effectively select the best solution for a specific domain. This approach was chosen based on extensive experiments that showed significant performance improvements when using meta-knowledge to select models tailored to specific types of network attacks. The observed results proved that no single classifier consistently outperformed others across all attack classes, justifying the need for a more adaptive meta-learning strategy. According to such idea, this paper proposes a novel ML-based scheme that can efficiently exploit a smart combination of machine learning algorithms and meta-features, taking advantage of the unique characteristics of the IDS domain.

Figure 1 shows the MIDES architecture, which, through a strategic combination of classifiers, and the adoption of advanced ensemble learning techniques, is able to overcome the shortcomings of individual classifiers while maintaining high performance. The design of the MIDES architecture was driven by the need to balance detection accuracy with computational efficiency. Experimental results show the benefits of combining multiple classifiers with the proposed multi-layer schema to overcome the limitations of single classifiers, in line with real-world scenarios where heterogeneous attacks occur.

MIDES works on different levels, ensuring that possible intrusions are identified and classified in the correct attack class. The system is easily extensible and robust in dynamic environments where attack patterns are constantly changing, since it is always possible to add new classifiers to the ensemble in order to face and correctly detect new threats. This extensibility ensures that the system remains effective and resilient against emerging threats without significant redesigns.

During the first phase, MIDES extracts the salient features of network packets through a constant analysis of traffic flow. The result of this preprocessing is a feature vector that is first fed to a binary classifier, named *Gatekeeper*, which performs an initial filtering, classifying the input feature vector as malicious or benign. The ideal classifier for this role has to satisfy two conflicting goals: on the one hand, it must try to avoid false negatives at all costs; on the other hand, if its classification were too strict, it would produce too many false positives, resulting in excessive waste of resources for the other classifiers in a vain attempt to classify harmless packets as if they were attacks. The Gatekeeper must also be extremely efficient, since it has the burden of analyzing all monitored network traffic. A classifier not optimized for this purpose would become the bottleneck of the entire system. In this work, the Gatekeeper classifier was selected based on its efficiency and strong recall, ensuring minimal false negatives and maintaining a high throughput for the whole system. This decision was further reinforced by the desire to handle large volumes of network traffic without introducing latency, ensuring that the system can operate in real-time, providing timely detection and response to network intrusions.

More formally, let $N^k$ denote the set of $k$ last observed network packets, and $\mathcal{F}$ represent the last $k$ observed data belonging to the vectorial space $\mathbb{F}$. The feature extraction process is represented as a function:

$$EF : N^k \rightarrow \mathbb{F}. \tag{1}$$

Then, let $\mathcal{G}$ represent the Gatekeeper classifier, and $\mathcal{C}_\mathcal{G}$ denote the classification decision made by the Gatekeeper. The Gatekeeper's classification process can be represented as:

$$\mathcal{C}_\mathcal{G} = f_\mathcal{G}(\mathcal{F}), \tag{2}$$

where $f_\mathcal{G}(\mathcal{F})$ is a function that maps the feature vector $\mathcal{F}$ to the classification decision $\mathcal{C}_\mathcal{G}$. The classification decision $\mathcal{C}_\mathcal{G}$ can label an incoming event as either benign ($e_b$) or malicious ($e_m$).

As a result of the Gatekeeper's action, an incoming event labeled as benign is quickly and efficiently evaluated and does not need to be analyzed by the other components of the system. Vice versa, an event labeled as malicious needs further investigation to verify that it is effectively an intrusion and, in that case, to appropriately classify the type of the ongoing attack. This two-tiered approach is consistent with best practices in the anomaly detection domain, where initial filtering reduces the load on more complex, resource-intensive analysis processes [16, 53]. The experimental evaluation presented here confirms that this hierarchical filtering significantly improves overall system efficiency and detection accuracy.

To perform this second classification, an ensemble of machine learning algorithms is adopted. This ensemble consists of $n$ classifiers, which are potentially quite different from each other and are remarkably effective at recognizing specific attack classes. Let $\mathcal{E} = \{\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_n\}$ represent the ensemble of classifiers. Each classifier $\mathcal{E}_i$, for $1 \leq i \leq n$, specializes in recognizing specific attack classes.

These classifiers are executed in parallel to analyze the event. Such a classification paradigm takes advantage of the higher classification capacity provided by a multitude of classifiers and thus achieves a better result that depends on the whole ensemble.

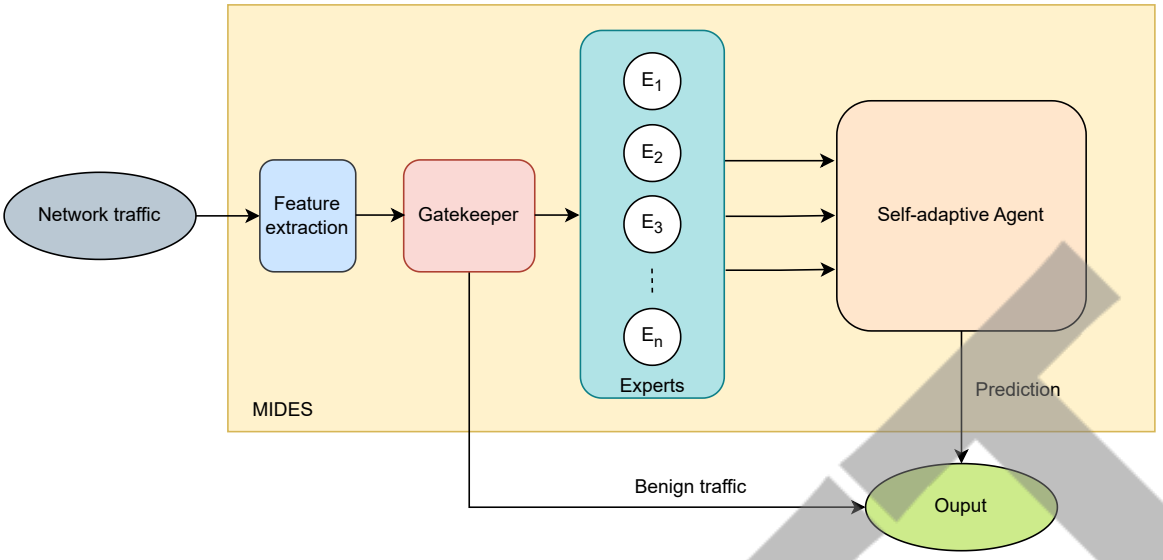



Figure 1: The multi-layer MIDES architecture.

The classification process in the second layer can be represented as:

$$\mathcal{C}_{\mathcal{E}} = \{\overline{\mathcal{E}_1(\mathcal{F}), \mathcal{E}_2(\mathcal{F}), \ldots, \mathcal{E}_n(\mathcal{F})}\}, \tag{3}$$

where $\mathcal{E}_i(\mathcal{F})$ represents the classification decision made by the i-th classifier $\mathcal{E}_i$ based on the feature vector $\mathcal{F}$ together with meta-information, and the horizontal bar indicates that the classifiers are executed in parallel. This vector of classification decisions and information such as the uncertainty of each classifier will then be exploited in different ways by the third tier (i.e., the self-adaptive agent) depending on the ensemble's *expertise* and *confidence* in recognizing specific attacks, as will be detailed below.

The selection of the classifiers to use can be tailored to the defense needs of the system to be protected. Given the ability to choose classifiers of any nature, this flexibility also paves the way for the use of IDSs in emerging network architectures and communication infrastructures that rely on the use of devices with limited computational resources, such as IoT networks. A realistic strategy could involve the use of classifiers known in the literature to be effective against specific classes of attack, and use as many classifiers as possible to better detect a multiplicity of attacks. Given the extreme dynamism, spread and evolution of attacks that undermine the security of IT infrastructures, it is necessary to think of a system that is ready to eventually integrate new classification systems. MIDES architecture can easily accommodate new classifiers to enrich the composition of the ensemble.

In order to determine the final classification, the output is analyzed by a *self-adaptive agent*. Let $\mathcal{S}$ represent the self-adaptive agent, and $\mathcal{C}_f$ denote the final classification decision. $\mathcal{S}$ exploits either a weighted voting module or a meta-classifier, based on a novel heuristic that takes into account both the *expertise* for that specific attack class and the *confidence* of the prediction made by these two macro-classifiers.

In the first case, the final output is the result of a weighted voting mechanism that takes into account the expertise of each classifier in recognizing specific attack classes. The final classification decision is determined as:

$$\mathcal{C}_f = f_{\mathcal{E}}(\mathcal{C}_{\mathcal{E}}), \tag{4}$$

where $f_{\mathcal{E}}(\mathcal{C}_{\mathcal{E}})$ is a function that takes the ensemble outputs $\mathcal{C}_{\mathcal{E}}$ and produces the final classification decision $\mathcal{C}_f$ using a weighted voting mechanism.

In the second case, the self-adaptive agent relies on an additional level of classification, using the technique of stacked generalization, exploiting a vector of meta-features derived from the ensemble output, $\mathcal{C}_{\mathcal{E}}$. Let $\mathcal{M}$ represent the meta-classifier. The final classification decision using stacked generalization can be represented as:

$$\mathcal{C}_f = f_{\mathcal{M}}(\mathcal{C}_{\mathcal{E}}), \tag{5}$$

where $f_{\mathcal{M}}(\mathcal{C}_{\mathcal{E}})$ is a function that takes the ensemble outputs $\mathcal{C}_{\mathcal{E}}$ and produces the final classification decision $\mathcal{C}_f$ using a meta-classifier based on stacked generalization. The final classification decision, $\mathcal{C}_f$, can thus be represented either by Equation 4 or Equation 5, depending on the decision of the self-adaptive agent $\mathcal{S}$. More generally, $\mathcal{C}_f$ can be expressed as follows:

$$\mathcal{C}_f = f_{\mathcal{S}}(\mathcal{C}_{\mathcal{E}}, \text{Expertise}(\mathcal{E}), \text{Confidence}(\mathcal{E})), \tag{6}$$

where Equation 6 can be reduced to either Equation 4 or Equation 5 depending on the *expertise* and *confidence* of the ensemble $\mathcal{E}$ with respect to the current event.

The remainder of this section discusses in detail the individual components of MIDES architecture.

### 3.1. Gatekeeper

The Gatekeeper $\mathcal{G}$ is the first line of defense of the system against malicious network traffic. The module consists of a single binary classifier that performs traffic filtering by labeling each record as benign or malicious. Since the classifier is specifically trained to minimize false negatives, the records classified as benign do not need further checks, and this greatly eases the computational burden of the following classification levels, which only have to analyze a small portion of the incoming traffic. Operationally, the Gatekeeper takes as input a preprocessed feature vector $\mathcal{F}$ extracted from a network flow, and returns a binary label: *benign* or *malicious*. The Gatekeeper is specifically designed to minimize the risk of misclassifying stealthy or low-profile attacks (e.g., SQL injection, infiltration, or brute-force web attacks) as benign. If the label is *benign*, the event is discarded and no further processing is applied. Otherwise, the event is passed to the ensemble of multi-class classifiers for detailed analysis. In the current implementation, the Gatekeeper uses a Decision Tree classifier trained with a standard entropy-based splitting criterion, selected as described below.

This binary filtering mechanism plays a critical role in maintaining the efficiency and reactivity of the entire system, as will be demonstrated in the experimental evaluation.

The main peculiarity of the Gatekeeper is the use of a classifier that enforces a very strict level of severity in order to promptly intercept incoming threats to the system. Consequently, the selection of a suitable classifier for the Gatekeeper should favor those that show high recall values and thus minimize the number of false negatives (i.e. attacks not recognized). The decision to enforce such firm behavior implies a trade-off: a high level of attack recognition often corresponds to a significant number of false alarms.

Such trade-off is carefully addressed by prioritizing high recall to ensure that no potential threats are missed, even at the cost of increased false positives. This is especially important for stealthy attacks that may exhibit benign-like patterns; the Gatekeeper is configured to err on the side of caution, forwarding such events for further analysis whenever there is any doubt. The reasoning is that false positives can be further analyzed and discarded in subsequent stages, while false negatives represent missed detections that could compromise system security. If a record is classified as potentially malicious by the Gatekeeper, it will simply be sent to the next levels of MIDES, to be further analyzed by expert classifiers that specialize in particular types of attacks.

The increased complexity introduced by the presence of the Gatekeeper is largely offset by the benefits it brings to the whole system. The first phase of binary classification enables highly effective and efficient filtering of input data, providing considerable robustness against attacks. Since benign records are extremely more common than those representing attacks in typical network traffic, the data to be analyzed in the second classification phase will be significantly less compared to the entire network traffic; this leads to a substantial reduction in the computational burden of the prediction performed by subsequent modules. By significantly reducing the volume of traffic that requires detailed analysis, the Gatekeeper increases the scalability of the IDS, ensuring that it can handle high traffic loads without performance degradation.

More formally, let $t_g$ be the time taken by the Gatekeeper $\mathcal{G}$ to classify an event, and $t_m$ be the time taken by the rest of the system when an event is labeled as malicious. Generally, $t_g \ll t_m$. Let $p_b$ and $p_m$ be defined as the fraction of events classified as benign or malicious by the Gatekeeper, respectively. The average time needed considering both benign and malicious classifications can be expressed as:

$$t_{avg} = p_b \cdot t_g + p_m \cdot (t_g + t_m). \tag{7}$$

Since $p_b + p_m = 1$, Equation 7 can be simplified as:

$$t_{avg} = t_g + p_m \cdot t_m. \tag{8}$$

Minimizing the number of events incorrectly classified as malicious by the Gatekeeper is thus crucial in reducing the average time needed for classification. Let $p_{TP}$ and $p_{FP}$ be respectively the fraction of true positives, i.e. events classified as malicious correctly, and false positives, i.e. events classified as malicious incorrectly, so that $p_m = p_{TP} + p_{FP}$.

Equation 8 can be rewritten as:

$$t_{avg} = t_g + (p_{TP} + p_{FP}) \cdot t_m. \tag{9}$$

To reduce the system response time and computational load, $p_{FP}$ should be minimized. As can be inferred from Equation 9, if MIDES analyzes $N$ events, the overhead (i.e., wasted time compared to the optimal system that always classify records correctly) can be computed as follows:

$$t_{overhead} = N \cdot p_{FP} \cdot t_m. \tag{10}$$

It is worth noting that the architecture of MIDES is extremely generalizable, considering that it allows any binary classifier to be used in the actual system. The best classifier to use may vary depending on the type of incoming traffic, and the optimal choice can only be made during the training phase.

Since the Gatekeeper analyzes all network traffic, a crucial requirement is that it be very fast in classification, as well as very accurate. However, it is not necessarily the case that the fastest classifiers are also the most accurate.

Let us define two generic metrics that will be used to evaluate potential classifiers, namely $\mu_{acc}$ as accuracy metric and $\mu_{time}$ as classification time metric. Given a set $C$ of $n$ candidate classifiers, the one chosen as the Gatekeeper

will have to satisfy two conflicting goals:

$$\begin{cases} \text{maximize } \mu_{acc} \\ \quad \text{and} \\ \text{minimize } \mu_{time}. \end{cases} \tag{11}$$

The common approach of using multi-attribute utility theory to maximize a single utility function that summarizes all the objectives is difficult to apply in this case. Therefore, a multi-objective trade-off analysis based on a Pareto dominance criterion was adopted in order to consider several conflicting objective functions, which are also characterized by non-comparable units of measure. The classifier selected as the Gatekeeper must be Pareto-optimal, that is, it must belong to the Pareto front. The use of Pareto front analysis ensures that the Gatekeeper can deliver optimal performance under the constraints of real-time traffic analysis.

Formally, the Pareto front is defined as a set $\Sigma$ of solutions $\sigma_i$ such that there is no other solution $v$ that dominates them, i.e., that is better than any $\sigma_i$ for both objectives. From a mathematical point of view, for the problem discussed here, this can be formulated as follows; the candidate classifier $c^* \in$ Pareto front if and only if $\nexists$ another classifier $c_i \in C$, $\forall i \in \{1, 2, \ldots, n\}$, such that:

$$\begin{cases} \mu_{acc}(c_i) \geq \mu_{acc}(c^*) \\ \quad \text{and} \\ \mu_{time}(c_i) \leq \mu_{time}(c^*) \\ \quad \text{and} \\ (\mu_{acc}(c_i) > \mu_{acc}(c^*) \quad \text{or} \quad \mu_{time}(c_i) < \mu_{time}(c^*)). \end{cases} \tag{12}$$

In other words, a solution $c^*$ is on the Pareto front if there is no other solution $c_i$ that is equal or better for both objectives, and strictly better for at least one of the objectives. In order to apply Pareto analysis to Gatekeeper selection, the candidate classifiers must be divided into two disjointed sets, dominated and non-dominated, in order to select a classifier from the non-dominated ones, as will be shown in the experimental section.

Furthermore, it is necessary to select a good metric for $\mu_{acc}$, which takes into account the specific characteristics of the Gatekeeper, and allows a good trade-off between false positives and false negatives, with particular attention to minimizing the latter. The metric adopted for this purpose is the $F_\beta$ score with $\beta = 2$ . The $F_\beta$ score is based on Van Rijsbergen's effectiveness measure [75], and is defined as:

$$F_\beta = (1 + \beta^2) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall}. \tag{13}$$

According to Van Rijsbergen, Equation 13 "*measures the effectiveness of retrieval with respect to a user who attaches $\beta$ times as much importance to recall as precision*". When $\beta = 2$, thus, the recall is given twice the importance as precision. This choice is motivated by the need to prioritize recall while still maintaining a reasonable balance with precision. The $F_2$ score's emphasis on recall helps to minimize the risk of false negatives, which is critical for the Gatekeeper's role in early threat detection. After several experiments, which will be described in detail, the classifier for the Gatekeeper was implemented using a decision tree with an entropy-based division criterion, as it has an excellent $F_2$ score and a high classification speed. The Decision Tree was chosen not only for its performance metrics, but also for its interpretability and speed. These characteristics are essential for real-time analysis, where decisions must be both fast and understandable for further tuning and improvement. Moreover, the Gatekeeper's strict filtering policy, combined with the use of data balancing techniques during training, ensures that rare and stealthy attack classes are adequately represented and detected. This conservative design significantly mitigates the risk of misclassifying stealthy attacks as benign at the first filtering stage.

This design-time selection of the Gatekeeper ensures that the classifier used at runtime offers an optimal trade-off between detection effectiveness and execution speed. Although the Gatekeeper does not adapt its behavior during execution, its preselection based on Pareto-optimality guarantees that it contributes to maximizing $\mu_{\text{acc}}$ and minimizing $\mu_{\text{time}}$ in practice.

Given the Gatekeeper's critical role in minimizing false negatives, especially for stealthy attacks that resemble benign traffic, we further investigated its behavior through a post hoc explainability analysis using SHAP (SHapley Additive exPlanations) [43]. This analysis focused on the rare instances where malicious samples were incorrectly classified as benign, and aimed to identify the specific features that contributed to such misclassifications. The findings are presented in Section 4.4.

### *3.2. Ensemble layer*

The filtering action performed by the Gatekeeper ensures a careful separation between benign and malicious records. However, this is not enough for the purposes of an IDS: due to the binary nature of the Gatekeeper, the predictions it obtains will only indicate the possible presence of an attack, without specifying what type of attack it is. Considering that defense procedures against intrusions can obviously be different depending on the type of attack, a second level of classification is necessary.

Since individual classifiers cannot detect all attack types with high accuracy, an ensemble of multiclass *expert* classifiers that work together is adopted in order to improve the overall effectiveness of the system. These classifiers are defined as experts because they are particularly accurate in classifying one or more attack types. Thus, the definition of this component requires identifying the capabilities of each expert classifier in order to select the most suitable ones. Expert classifiers are selected based on their

ability to detect specific types of attacks. This specialization ensures that each classifier can contribute its unique strengths to the ensemble.

Specifically, a classifier is defined to be expert with respect to a subset of attack classes when it showed evidence of consistently maintaining performance above an appropriate threshold as a result of several experiments.

More formally, let $A = \{a_1, a_2, ..., a_m\}$ be a set of $m$ attack classes and $C = \{c_1, c_2, ..., c_n\}$ a set of $n$ classifiers. A classifier $c_j$ is expert in classifying an attack class $a_i$ if, given a metric $\mu_{\mathcal{E}}$ and a threshold $\tau$, the following property holds:

$$\mu_{\mathcal{E}}(a_i, c_j) \geq \tau. \tag{14}$$

In this work, the metric $\mu_{\mathcal{E}}$ adopted is the $F_\beta$ score with $\beta = 1$, as defined in Equation 13. This corresponds to the classical $F_1$ score, which gives equal importance to precision and recall. Moreover, $\tau = 0.8$ has been experimentally set as a parameter to identify expert classifiers for specific attack classes. This threshold was chosen based on empirical analysis, which indicated that classifiers meeting this criterion consistently provided reliable performance across different attack scenarios.

It would be desirable to identify a group of experts that are different enough from each other so as to cover all possible attack classes; the ensemble layer will use these experts, exploiting their specific capabilities in order to avoid the occurrence of critical classes (i.e., subsets of the attack classes against which the performance of the ensemble is significantly lower than usual). MIDES is versatile enough to support the use of heterogeneous supervised classifiers. The choice of the best classifiers depends on the application scenario and the expected type and frequency of attacks. In this work, the experimentally selected set of expert classifiers for the ensemble consists of four classifiers, namely Decision Tree, Neural Network, Random Forest and Naive Bayes. The inclusion of diverse classifiers in the ensemble offers several advantages, such as increased resilience to different types of attacks and the ability to leverage complementary strengths. For example, while Decision Trees may excel in interpretability, Neural Networks can provide high accuracy for complex patterns, and Random Forests can provide greater robustness through ensemble averaging.

After identifying the expert classifiers, it is necessary to determine how to merge their outputs. This task is carried out by a self-adaptive agent, which will appropriately choose between two possible techniques, i.e., weighted voting mechanism performed by the experts, or an additional level of meta-classification. As will be described in the following, the choices made by the self-adaptive agent depend on the context and specific characteristics of the incoming traffic. This ensures that the system can maintain high performance over time.

### *3.3. Self-adaptive agent*

This section presents the self-adaptive agent, first describing the two modules of weighted voting and meta-classification, and then explaining the behavior of the *arbiter* who chooses which of the two modules should be used for the current input event, based on heuristics. The self-adaptive agent is designed to dynamically adapt to evolving network conditions and attack patterns. An overview of the self-adaptive agent shown in Figure 2.

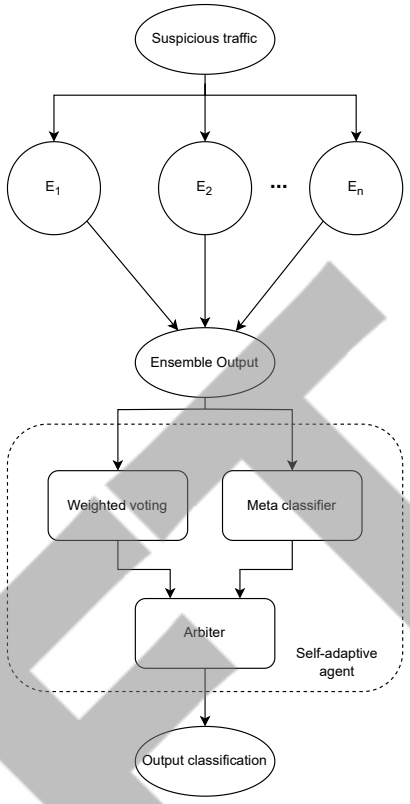



Figure 2: Ensemble of experts and self-adaptive agent.

#### *3.3.1. Weighted voting*

The weighted voting module assigns different levels of importance to each classifier's decision based on its accuracy. These weights could then be used to obtain the overall ensemble vote, represented by the final label given to the input record. This allows the ensemble to make more informed decisions, exploiting the strengths of each classifier while mitigating their individual weaknesses.

The simplest approach is to adopt weights directly proportional to the overall accuracy of classifiers. However, single classifiers are often more reliable at identifying some specific attack classes and less proficient at discriminating others. Assigning a single weight to each classifier is therefore too restrictive, and does not reflect their true capabilities, leading to a consequent downgrading in the overall predictive performance of the system. To address this limitation, MIDES approach will involve assigning multiple weights to each classifier, as discussed below.

Let us first consider a simple ensemble of classifiers denoted by $\mathcal{C}$, where each classifier $\mathcal{C}_i$, for $1 \leq i \leq n$ contributes to the overall classification. The weight associ-

ated with each classifier, denoted as $w_i$, represents its significance in the ensemble. A simple approach might define the classification decision of the ensemble as the weighted sum of individual classifier decisions, as follows:

$$\mathcal{D}^{\mathcal{W}} = \sum_{i=1}^{n} w_i \cdot \mathcal{D}_i, \tag{15}$$

where $\mathcal{D}_i$ represents the classification decision made by classifier $\mathcal{C}_i$.

Let us consider a specific classifier, denoted as $\mathcal{C}_r$, which is deemed as very reliable, and thus has a significantly higher weight compared to others. For the NFL theorem, there will be some attack classes that are not adequately recognized by $\mathcal{C}_r$. Consequently, the entire system may fail, as the combined votes of all other classifiers will have a lower weight. This scenario can be formalized by defining a subset of attack classes, denoted by $A_p$, for which $\mathcal{C}_r$ performs poorly. The classification decision made by the ensemble for these attack classes will be influenced heavily by the incorrect decisions of $\mathcal{C}_r$. Since the overall classification decision can be represented as

$$\mathcal{D}^{\mathcal{W}} = w_r \cdot \mathcal{D}_r + \sum_{i \neq r} w_i \cdot \mathcal{D}_i, \tag{16}$$

it is clear that when $w_r > \sum_{i \neq r} w_i$, the final decision is heavily affected by the classifier $\mathcal{C}_r$, even if it makes a mistake. In practice, in many cases there will be a subset of dominant classifiers that will heavily affect the performance of the whole system, both positively and negatively. Consequently, there will be a propagation of the *weaknesses* of such classifiers on the final prediction, affecting the overall performance of the ensemble, preventing it from outperforming individual classifiers.

By assigning weights based on the classifiers' performance on specific attack classes, the system can better leverage their strengths and compensate for their weaknesses. This approach ensures that no single classifier has a disproportionate influence on the final decision, resulting in more accurate and reliable intrusion detection. Thus, the system needs more fine-grained control of the weights: this can be achieved by exploiting the concept of expert classifier discussed earlier. Every classifier will be assigned a vector of weights, where each element represents the accuracy of the learning model with respect to a specific class of attacks.

More formally, let $w_{ij}$ be the weight of the $j$-th classifier for the $i$-th attack class. Given an input record $x$, the probabilities predicted by each classifier $j$ for each possible attack class are denoted as:

$$p_{c_j}(\mathcal{D}_j = a_i | x), \tag{17}$$

for $i = 1, 2, ..., m$ and $j = 1, 2, ..., n$.

The vector of weighted predictions produced by the expert $j$ is defined as follows:

$$\boldsymbol{P}_{c_j}(x) = [\, p_{c_j}(a_i|x) \cdot w_{ij} \,], \tag{18}$$

for $i = 1, 2, ..., m$ and $j = 1, 2, ..., n$.

The overall vote that the ensemble assigns to each attack class $i$, given the input record $x$, is defined as follows:

$$v_i(x) = \sum_{j=0}^{n} \boldsymbol{P}_{c_j}(x)[i], \tag{19}$$

for $i = 1, 2, ..., m$, and where $\boldsymbol{P}_{c_j}(x)[i]$ indicates the $i$-th element of the vector $\boldsymbol{P}_{c_j}(x)$.

Finally, the ensemble prediction is given by the class with the maximum weighted vote, as follows:

$$\mathcal{D}^{\mathcal{W}} = \underset{i \in [1,...,m]}{\arg\max} \{v_i(x)\}. \tag{20}$$

Using specific weights for each $< a_i, c_j >$ pair makes it possible to overcome the limitations highlighted above. Such fine-grained weights do not result in dominant classifiers if the experts are carefully chosen to compensate for each other's weaknesses and not incur in critical classes. Such an approach ensures that the ensemble can handle a wide range of attack scenarios.

Selecting an appropriate metric to calculate the weights is critical, since an improper choice could cause a degradation of system performance. For example, suppose the recall values of a classifier against each class are chosen as weights. In this case, a naive classifier that always predicts the same class would get maximum weight for that class, heavily influencing ensemble decisions every time the real class is a different one. The problem is caused by the fact that the recall metric does not take false positives into account.

A better metric must take into account both precision and recall. For this reason, the selected metric is $F_1$ score, which gives equal importance to both. Such a choice ensures that classifiers with high false-positive rates do not excessively influence the ensemble's decisions. This balance is essential for maintaining high detection accuracy while minimizing false alarms.

This means that if the classifier has received a high weight for a particular attack class, it is not only effective in recognizing true positives in that class, but also in minimizing false positives; thus, it is unlikely that such a classifier will mistake a critical class for an attack on which the classifier is an expert. The choice of this metric was validated through extensive testing, demonstrating its effectiveness in reflecting the true performance of each classifier in the ensemble.

#### *3.3.2. Stacked generalization and meta classifier*

Stacked generalization, as originally introduced in [78], is a scheme for minimizing the generalization error of a classifier or ensemble of classifiers. The goal is to reduce the bias of individual classifiers (base level) by training an additional classifier (meta level) with a new set of meta-features that represent information provided by the predictions of individual base classifiers. This technique allows

the system to make more accurate predictions by combining the outputs of base classifiers through an additional layer of learning.

Let $X$ represent the input dataset consisting of incoming network events, and let $Y$ be the corresponding labels. Let us consider an ensemble of base classifiers, denoted as $C$, where each base classifier $C_i$, for $1 \leq i \leq n$, produces individual classifications for the input records. During the training phase, each base classifier $C_i$ is trained on all available data. The *meta classifier*, denoted as $\mathcal{M}$, is then trained on the resulting predictions of the base classifiers. The training of the meta classifier can be formulated as a supervised learning problem, where the features are predictions made by the base classifiers $C_i$, and the target is predicting the true labels $Y$.

The meta classifier learns the relationships between the individual classifiers' predictions and $Y$ by exploiting a cross-validation strategy. This allows the identification of patterns and correlations that may not be apparent from the individual base classifiers alone. In the prediction phase, new input records are classified by the base classifiers, and the individual classifications are then examined by the meta classifier $\mathcal{M}$. If $\mathcal{D}_i(x)$ denotes the classification decision made by base classifier $C_i$ for input record $x$, the classification decision made by the meta classifier for the predictions $\mathcal{D}_1(x), \mathcal{D}_2(x), \ldots, \mathcal{D}_n(x)$ of the base classifiers can be expressed as follows:

$$\mathcal{D}^{\mathcal{M}}(\mathcal{D}_1(x), \mathcal{D}_2(x), \ldots, \mathcal{D}_n(x)). \tag{21}$$

The advantage of this approach is that the discovery of relationships between individual classifiers and labels is automatically done by the algorithm during the training phase. This allows the overall system to produce a more accurate prediction.

To fully define the *meta classifier*, two main problems must be solved:

- identifying the set of meta features that will be given as input to the meta classifier;
- choosing the best possible meta classifier to maximize the overall performance of the system.

The simplest method for choosing the set of meta-features is to use only the labels predicted by the individual base classifiers. This technique is very simple and efficient to implement, but it does not allow the meta-classifier to generalize as it should. Indeed, by exploiting such a limited set of meta-features, the base classifications heavily influence the final prediction of the meta-classifier. Therefore, the final classification would be strongly influenced by the accuracy with which the individual experts predict the class and not by other factors, such as the uncertainty of the original classifier about the prediction itself.

A simple improvement in the choice of meta-features is thus to add, to each base level classification, the confidence of their prediction. Such a choice allows the output of the base classifiers to be evaluated according to their certainty, resulting in more reliable and accurate final predictions by learning which classifiers are more confident and for which attack classes their output has a higher probability of being correct. This approach helps to mitigate the impact of uncertain or less reliable base classifier predictions.

In this regard, the authors of [72] proposed adding the probability distribution extracted from the individual base classifiers as a set of meta-features. This probability distribution consists of the probability that each classifier associates with each predicted class for that particular record.

In [20], it was shown how using additional meta-features can further improve the performance of the final system. These additional meta-features provide valuable information about the distribution and reliability of the base classifiers' predictions, enabling the meta-classifier to make more informed decisions. In particular, the authors of [20] proposed to use the entropy of the probability distribution and an additional feature consisting of the product between the probability distribution and its maximum.

With these results in mind, MIDES adopts the following set of features, given an input record $x$:

- the probabilities predicted by each base classifier for each possible attack class, as presented in Equation 17;
- the entropy of each probability distribution:

$$E_{c_j}(x) = -\sum_{i=1}^{m} p_{c_j}(a_i|x) \cdot \log_2 p_{c_j}(a_i|x), \tag{22}$$

  for $j = 1, 2, ..., n$;
- each probability distribution multiplied by the weight of the j-th classifier for the i-th attack class, as defined in Equation 18.

The amount of meta-features depends on the number of base classifiers, $n$, and the number of attack classes, $m$. Thus, the cardinality of the set of meta-features is $n(2m+1)$.

It may seem redundant to use both $p_{c_j}(a_i|x)$ and $\boldsymbol{P}_{c_j}(x)$ as meta-features. However, just replacing $p_{c_j}(a_i|x)$ with $\boldsymbol{P}_{c_j}(x)$ would not lead to satisfactory results. Džeroski and Ženko [20] shows that using only probability distributions (or some variation thereof, replacing probability distributions) for stacking produces final classifiers that do not perform any better than simply selecting the best classifier of the ensemble by cross-validation. Instead, the combined use of $p_{c_j}(a_i|x)$ and $\boldsymbol{P}_{c_j}(x)$ yields an information gain regarding the accuracy of the individual classifiers for each predicted attack class, significantly improving the overall performance of MIDES.

The last design choice regards the selection of the classification algorithm to be employed. As with the Gatekeeper and the ensemble of classifiers, MIDES is general enough to allow for the use of any supervised classifier.

The best choice, in any case, always depends on the scenario in which the system will be used and on the type of data it will receive as input.

Here, a Decision Tree with selection criteria based on the Gini index has been adopted as the meta-classifier. The advantages deriving from this choice are manifold: ease of use, fast training speed, high prediction accuracy and clarity in understanding the individual classifications given the *white box* approach of Decision Trees. This choice is further validated by comparative experiments with other techniques, which highlight the robustness of the Decision Tree in producing accurate final classifications.

### *3.3.3. Arbiter*

To further refine the effectiveness of MIDES in detecting a multiplicity of attacks, the output of the voting and meta-classifier modules is ultimately handed over to a decision-making module known as the Arbiter. The Arbiter enhances the decision-making process by dynamically selecting the module (voting or meta-classifier) that provides the most reliable prediction for each input record. This adaptive approach ensures that the system can respond to different attack scenarios while maintaining high accuracy.

Although the accuracy of the voting and meta-classifier modules is already high, a system that is able to analyze the outputs of both and decide which is the more reliable of the two for the current input record can achieve even better results.

The rationale underlying the self-adaptive agent is to exploit both a static and a dynamic term to evaluate, respectively, the expertise of the macro-classifier (i.e., voting or meta-classifier) and its confidence in the prediction. This dual perspective allows the Arbiter to make more informed and balanced decisions. In particular, the dynamic component is based on a measure of uncertainty, enabling the Arbiter to adapt its decision according to the reliability of each prediction.

Given an input record $x$ and the output of the two macro-classifiers, the following heuristic score is calculated for both:

$$s(x) = \gamma \cdot \mathit{Expertise}(x) + (1 - \gamma) \cdot \mathit{Confidence}(x), \quad (23)$$

where $\gamma$ is a parameter between 0 and 1 which is used to assign the relative importance of the two terms. Low values of $\gamma$ give greater importance to the dynamic part of the formula (confidence), while values of $\gamma$ close to 1 give greater importance to the static term (expertise). Both expertise and confidence are represented by values between 0 and 1, so the whole heuristic score $s$ is likewise a value between 0 and 1. After computing the $s$ score for both macro-classifiers, the self-adaptive agent will choose the one which received the highest score. This approach ensures that the final decision leverages the strengths of both modules and dynamically adapts to the nature of the input data.

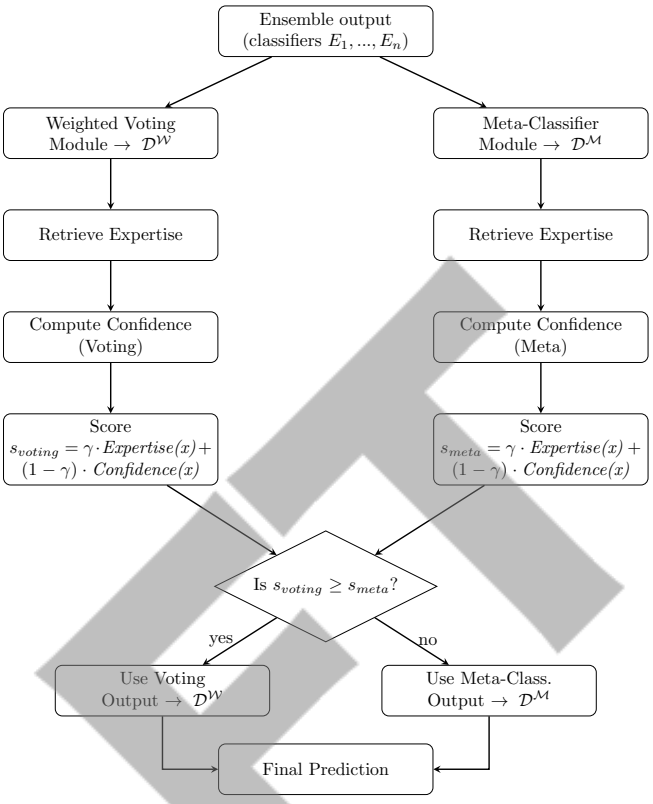



Figure 3: Flow diagram illustrating the Arbiter's dynamic strategy selection based on confidence, uncertainty, and retrieved expertise values.

As the names of the terms suggest, the *expertise* seeks to estimate how reliable a particular macro-classifier is in detecting a specific attack, while the *confidence* depends on how certain the classifier is of the output returned. A low confidence value indicates that the classifier is in doubt between two or more classes, while a high value indicates that the classifier is certain of its prediction.

The *expertise* is somewhat static, as it can be quickly estimated from values computed during the training phase, while the *confidence* is extremely dynamic, as it depends on the specific input record as well.

The metrics chosen to calculate expertise and confidence are derived from values previously discussed. However, it is necessary to extend to macro-classifiers some of the definitions already given for weak classifiers.

The *expertise* is calculated similarly to the weights assigned to the individual expert classifiers of the ensemble, using the $F_1$ score as a metric to evaluate, during the training phase, the performance of the two macro-classifiers (voting and meta-classifier). As discussed earlier, the choice of the $F_1$ score is important to assign equal importance to false positives and false negatives, exploiting precision and recall.

As for the dynamic term, let us define the confidence of a macro-classifier in its prediction as:

$$\mathit{Confidence}(x) = 1 - E_c(x), \quad (24)$$

**Algorithm 1** Runtime classification flow of MIDES.

**Require:** feature vector $\mathcal{F}$
**Ensure:** final decision $\mathcal{D}_f$
$\mathcal{D}_\mathcal{G} \leftarrow \textsc{Gatekeeper}(\mathcal{F})$ ▷ Gatekeeper decision
**if** $\mathcal{D}_\mathcal{G} = benign$ **then**
  **return** *benign* ▷ early exit on benign traffic
**end if**
**for all** expert classifier $\mathcal{E}_j \in \mathcal{E}$ **in parallel do**
  $\boldsymbol{P}_{c_j} \leftarrow \mathcal{E}_j(\mathcal{F})$ ▷ probability vector for $m$ classes
**end for**
$\mathcal{D}^\mathcal{W} \leftarrow \textsc{WeightedVoting}\big(\{\boldsymbol{P}_{c_j}\}_{j=1}^n\big)$
$\mathcal{D}^\mathcal{M} \leftarrow \textsc{MetaClassification}\big(\{\boldsymbol{P}_{c_j}\}_{j=1}^n\big)$
$s^\mathcal{W} \leftarrow \textsc{HeuristicScore}\big(\mathcal{F}, \mathcal{D}^\mathcal{W}\big)$
$s^\mathcal{M} \leftarrow \textsc{HeuristicScore}\big(\mathcal{F}, \mathcal{D}^\mathcal{M}\big)$
**if** $s^\mathcal{W} \geq s^\mathcal{M}$ **then** ▷ Arbiter decision
  $\mathcal{D}_f \leftarrow \mathcal{D}^\mathcal{W}$
**else**
  $\mathcal{D}_f \leftarrow \mathcal{D}^\mathcal{M}$
**end if**
**return** $\mathcal{D}_f$

where $E_c(x)$ is the entropy of the prediction vector, similarly to Equation 22. This formulation corresponds to an uncertainty-aware confidence estimation, in line with established uncertainty modeling techniques. In this context, the entropy $E_c(x)$ quantifies the level of dispersion in the prediction vector: a low-entropy distribution implies a confident prediction, whereas high entropy indicates uncertainty across multiple classes.

Regarding the voting module, entropy is calculated on the vector consisting of all votes $v_i$ as they were defined in Equation 19, after being properly normalized. As for the meta-classifier, entropy is calculated on the probability distribution returned by the classifier itself. Algorithm 1 outlines the runtime classification flow of MIDES, from Gatekeeper filtering to the final decision taken by the self-adaptive agent, while the flow diagram shown in Figure 3 summarizes the overall decision-making process of the Arbiter.

## 4. Experimental Evaluation

This section presents a comprehensive experimental evaluation of MIDES and its components. First of all, the experimental setting is defined, describing the dataset used and the metrics adopted. Then, a comparison between different classifiers is performed to select the most suitable one for the Gatekeeper role, together with the evaluation of the expert classifiers used in the second layer of MIDES. Moreover, the system is compared with other state-of-the-art systems on two different datasets to demonstrate its capabilities both in terms of accuracy and classification speed. Finally, additional experiments show that the Gatekeeper component significantly reduces execution times, improving the overall efficiency.

### *4.1. Experimental setting*

When evaluating the performance of an Intrusion Detection System, choosing an appropriate dataset that is as realistic as possible is critical.

During the last two decades of research on Intrusion Detection Systems, various datasets have been released by the academic community to evaluate the performance of proposed solutions. A systematic review of the main datasets released is provided in [59, 18]. One of the earliest and widely used datasets is KDD98, which was released in 1998 by DARPA (Defense Advanced Research Project Agency). This dataset collects nearly 5 million records with 41 features and labels such as normal and abnormal and covers 2 months of TCP dump packets. Although it is considered one of the earliest dataset development efforts, its use has been widely criticized for issues of accuracy and poor ability to reflect real-world conditions [19, 46, 2].

The same dataset was later used as the basis for the Third International Knowledge Discovery and Data Mining Tools Competition dataset, commonly called the KDD Cup99. Both datasets are considered dated today, because many of the modern attacks are not addressed, but they nevertheless remain widely used to compare modern solutions with older systems.

A statistical analysis reported in [69] performed on the KDD Cup99 dataset also showed that many of the samples collected (approximately between 75% and 78%) are duplicates, so the learning models evaluated on that dataset may have been affected by this issue. For this reason, the authors of the analysis created a dataset called NSL-KDD in 2009, which solves the problems of KDD Cup99 by eliminating duplicate records. CAIDA is a dataset that contains traffic traces that relate exclusively to Distributed Denial-of-Service (DDoS) attacks.

In [66], the authors propose a systematic approach to dynamically generate datasets which not only reflect the traffic compositions and intrusions, but are also modifiable, extensible, and reproducible. Through this approach, the authors released ISCX 2012 which contains different types of attacks such as DoS, DDoS, and SSH brute force.

One of the most recent and widely used datasets is CIC-IDS2017 [65]. This dataset was created by the Canadian Institute for Cybersecurity on a simulated environment with 12 victim virtual machines and a network of attackers over a 5-day time frame, where at specific time intervals various attacks were carried out.

Given the presence of such a wide multitude of attacks and the relatively recent release compared to other datasets, CIC-IDS2017 has been selected for the experimental evaluation of MIDES. The dataset includes 15 types of traffic, with one representing normal traffic and the other 14 representing various types of attacks. Additionally, the CSE-CIC-IDS-2018 dataset [65] is also used, as it provides updated attack patterns and more complex traffic scenarios. These datasets, expecially in their improved versions [41], present a diverse range of attack types

and are representative of real network traffic, making them suitable for assessing the robustness and adaptability of MIDES.

In order to perform the experimental evaluation of MIDES, 80 relevant features have been selected for both datasets, excluding the features that contain information about the source and destination IP addresses, timestamp, and flow ID. The flow ID is a string that identifies the flow by concatenating the source and destination IP addresses, the source and destination ports, and the communication protocol. Excluding these features enables the classifier to better distinguish between malicious and benign traffic. The rationale is that including the source and destination IP addresses could lead the classifier to label a specific IP address as the source or destination of malicious traffic, resulting in further misclassifications, while the flow ID contains redundant information. Similar preprocessing steps were taken for both datasets, to ensure balanced representation of all attack classes.

Since some of the attack classes, like Heartbleed, Infiltration, and Web Attack w/ Sql Injection have few records, an oversampling mechanisms has been adopted to re-balance the datasets. The Gatekeeper and expert training phases adopted the SMOTE oversampling technique [7]. The classes involved were oversampled so that at the end of the process they were represented by 20,000 records each. This allowed the experts to more accurately classify the records in the original dataset, thus improving both the voting stage and the creation of the meta-dataset. Then, once the meta-dataset was created, random oversampling was performed to increase the generalization capacity of the meta-classifier.

To train and test all models, all the traffic recording sessions in each dataset have been merged and then the resulting dataset was randomly divided into 80% for training and 20% for testing purposes.

All the experiments presented in the following sections have been run 1000 times using different train and test sets each time. The following tables and figures report the average of the obtained results. All tests were performed on a VM equipped with Intel(R) Xeon(R) Gold 6242 CPU @ 2.80GHz, 32GB of RAM and NVIDIA GRID A100 GPU.

### *4.2. Metrics*

One of the most commonly used metrics to evaluate IDSs is average accuracy, defined as:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}, \tag{25}$$

where TP, TN, FP, and FN are the true positives, true negatives, false positives, and false negatives, respectively. However, accuracy alone is not sufficient to evaluate different approaches, as data are often skewed towards the more common classes.

To provide a more detailed analysis of MIDES performance, additional metrics such as precision (positive predictive value) and recall (sensitivity) are used, respectively, as a measure of fidelity and completeness. These are defined as follows:

$$\text{Precision} = \frac{TP}{TP + FP}, \tag{26}$$

$$\text{Recall} = \frac{TP}{TP + FN}. \tag{27}$$

The $F_1$-score, in turn, is defined as the harmonic mean of precision and recall. As mentioned in Section 3.1, a variant of this metric, called the $F_2$-score, is considered for the evaluation. Both the $F_1$-score and the $F_2$-score are derived from the formula of the $F_\beta$-score (Equation 13), with $\beta = 1$ and $\beta = 2$, respectively:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}, \tag{28}$$

$$F_2 = 5 \cdot \frac{precision \cdot recall}{4 \cdot precision + recall}. \tag{29}$$

The $F_2$-score gives recall twice the importance compared to precision, and it will be used to select the best classifier to act as the Gatekeeper.

In such an analysis, both the macro average F1-scores and the weighted F1-scores are presented to provide a comprehensive assessment of the model's performance. The macro average F1-score is simply the mean of the F1-scores across all classes, while the weighted F1-score accounts for the frequency and importance of each class in the dataset. The macro F1-score provides an overall view of the model's performance but may not adequately reflect the importance of less frequent attack classes. In contrast, the weighted F1-score makes it a more informative choice in contexts with a high class imbalance. However, this approach can mask performance on minority groups, such as attacks, in favor of majority classes like benign records. Therefore, presenting and discussing both scores allows for a detailed evaluation of the models' effectiveness.

To further enrich the evaluation in imbalanced settings, we also report two additional metrics: the Area Under the Precision-Recall Curve (AUC-PR) and the Matthews Correlation Coefficient (MCC). AUC-PR is particularly suited for imbalanced datasets, as it emphasizes performance on the positive class and reflects the trade-off between precision and recall across different thresholds. A high AUC-PR indicates strong detection capabilities even when positive instances (i.e., attacks) are rare.

The Matthews Correlation Coefficient (MCC), on the other hand, provides a balanced measure that considers all four entries of the confusion matrix (TP, TN, FP, FN). It returns a value in the range $[-1, 1]$, where 1 indicates perfect prediction, 0 corresponds to random classification, and -1 indicates total disagreement between prediction and ground truth. The MCC is defined as follows:

$$\text{MCC} = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}. \tag{30}$$

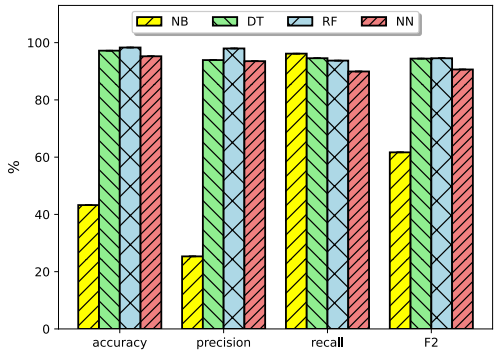



Figure 4: Comparison of candidate Gatekeeper models.

This metric is particularly informative in binary classification tasks, where both false positives and false negatives contribute to the overall quality of the decision process. Although the Gatekeeper prioritizes minimizing false negatives, MCC offers a more balanced view of classification performance that complements recall-focused metrics like $F_2$.

### *4.3. Gatekeeper selection*

As previously discussed, the filtering action performed by the Gatekeeper is essential for the thorough recognition of attacks in MIDES. This filtering also significantly increases the efficiency of MIDES by quickly discarding clearly benign traffic using a binary classifier, and forwarding only potentially malicious samples to the second level, i.e., the ensemble of expert classifiers.

To select the most suitable model for this task, four candidate classifiers have been evaluated on the CIC-IDS-2017 dataset: Naive Bayes (NB), Decision Tree (DT), Random Forest (RF), and Neural Network (NN). The dataset was rebalanced using SMOTE to ensure adequate representation of rare and stealthy attack classes, thus minimizing the risk of bias in the evaluation. The selection was guided by a Pareto-front analysis based on two criteria: $F_2$-score (to prioritize recall and minimize false negatives) and classification time (to ensure low latency). Among the non-dominated classifiers, the Decision Tree offered the best trade-off between effectiveness and efficiency and was therefore selected as the Gatekeeper for MIDES.

Figure 4 and Table 3 show the classification results and timing for each model, while Table 2 summarizes the parameters used for each classifier. The selection of these parameters was based on preliminary experiments that indicated their effectiveness in balancing performance and computational efficiency.

Figure 4 shows the accuracy, precision, recall and $F_2$-score obtained by the four systems. As can be seen, the results yielded by Naive Bayes are low for almost all metrics. The only exception is the recall obtained by Naive Bayes, which is surprisingly good. This demonstrates that recall alone is not a sufficient metric, because even low-quality classifiers can achieve high recall values, at the expense of precision. This is also reflected in the low MCC (0.35) and AUC-PR (0.45) of Naive Bayes, confirming its inability to balance false positives and false negatives effectively. The Neural Network achieves better results, but it is slower than others in classification time. The fact that the Neural Network is faster than Random Forest during training can be explained by its greater ability to benefit from GPU acceleration. Neural Networks rely on operations which are highly parallelizable and optimized for GPU hardware. In contrast, Random Forests are less suited to GPU parallelization and rely more on sequential or CPU-bound computations. As discussed earlier, for the Gatekeeper selection, the classifiers are divided into two disjoint sets, i.e., dominated and non-dominated, in order to choose one of the non-dominated ones. The $\mu_{acc}$ metric to maximize is the $F_2$-score, while the $\mu_{time}$ metric to minimize is the classification time.

As Fig. 4 and Table 3 show, the only dominated classifier is the Neural Network, as there are other classifiers that perform better for both metrics.

The other three classifiers are not-dominated. However, even if NB has the fastest training time, it was discarded as unsuitable, given its poor performance. The Gatekeeper's task, in fact, is extremely sensitive: anything that is labeled as benign by this classifier will no longer be checked by the rest of the system, for efficiency reasons, so it is important that the Gatekeeper is very accurate. This is particularly critical for stealthy attacks such as SQL injection and brute-force web attacks, which can exhibit patterns similar to benign traffic. The use of the $F_2$ score during selection helps ensure these challenging cases are detected and not misclassified.

Decision Tree and Random Forest present excellent re-

Table 2: Parameters used for the Gatekeeper selection.

| **System** | **Notes and parameters** |
|---|---|
| Naive Bayes | Gaussian NB |
| Decision Tree | Maximum depth: 15,<br>split criterion: information gain |
| Random Forest | 10 estimators, with max depth of 15 |
| Neural Network | 3 layers. Input: 128 neurons with ReLU,<br>output: 2 neurons with softmax |

Table 3: Detailed performance metrics of Gatekeeper candidate models.

| | **NB** | **DT** | **RF** | **NN** |
|---|---|---|---|---|
| Accuracy | 43.26% | 97.20% | **98.30%** | 95.24% |
| Precision | 25.34% | 93.91% | **97.97%** | 93.53% |
| Recall | **96.16%** | 94.56% | 93.73% | 89.93% |
| $F_2$-score | 61.68% | 94.43% | **94.55%** | 90.63% |
| AUC-PR | 0.45 | **0.95** | 0.94 | 0.91 |
| MCC | 0.35 | **0.89** | 0.88 | 0.85 |
| Training time | **3.13s** | 78.71s | 251.52s | 101.45s |
| Classification time | 0.92s | **0.19s** | 5.78s | 8.15s |

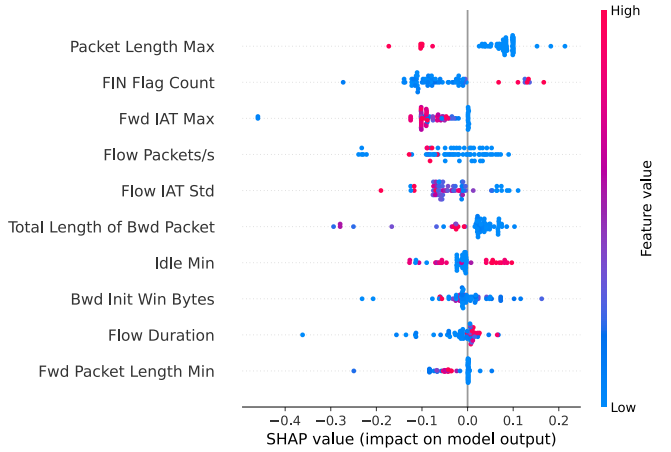



Figure 5: SHAP summary plot for false negatives in Gatekeeper classification.

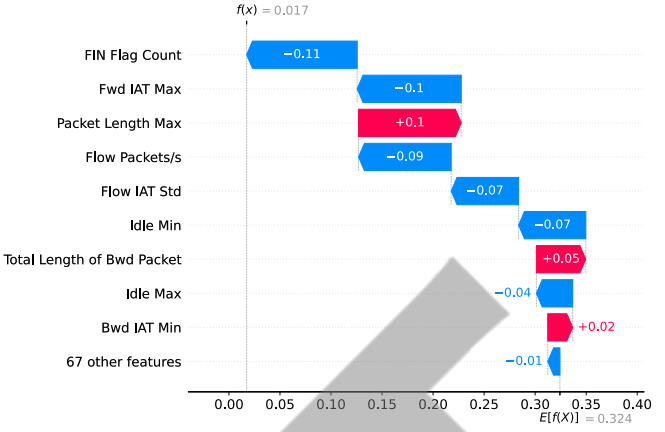



Figure 6: SHAP waterfall plot for one false negative sample. Features with the largest contributions to the classification decision are shown.

sults, and perform very closely with each other. Random Forest scored higher for precision and Decision Tree performed slightly better for recall, while the $F_2$-score is essentially the same for the two systems. Their MCC and AUC-PR values are also comparable (RF: 0.88 and 0.94, DT: 0.89 and 0.95), confirming that both classifiers are strong performers even under imbalanced conditions.

Given this similarity, the deciding factor became their training and classification times, to ensure greater efficiency. As can be seen in Table 3, the Decision Tree is faster than Random Forest both in training and classification. Given its efficiency and comparable $F_2$-score, the Decision Tree was ultimately selected as the Gatekeeper for MIDES. Notably, the selected Decision Tree classifier also achieves one of the highest AUC-PR and MCC values among the candidates, reinforcing its suitability for this critical role.

### *4.4. Explainability analysis of Gatekeeper misclassifications*

To better understand the behavior of the Gatekeeper and investigate its rare misclassification cases, we conducted a post hoc explainability analysis using SHAP (SHapley Additive exPlanations) [43]. Our focus was specifically on *false negatives*, i.e., malicious events that the Gatekeeper mistakenly labeled as benign.

SHAP provides a feature-level explanation of each prediction by quantifying how much each feature contributes to increasing or decreasing the predicted output. For this analysis, we trained the Gatekeeper on the CIC-IDS-2017 dataset and collected all misclassified malicious samples from the test set.

Figure 5 shows a SHAP summary plot for the misclassified instances, where each point represents the SHAP value of a feature for a specific sample. The plot highlights the features that most frequently influenced the Gatekeeper's misclassifications. For instance, `Packet Length Max`, `FIN Flag Count`, and `Fwd IAT Max` appear prominently and repeatedly, suggesting that specific feature ranges may lead the classifier to underestimate the likelihood of an attack.

Table 4: Top 5 most influential features in false negatives.

| **Feature** | **Freq** | **Mean** \|SHAP\| | **Std** \|SHAP\| |
|---|---|---|---|
| Packet Length Max | 69 | 0.086 | 0.031 |
| Total Length of Bwd Packet | 60 | 0.054 | 0.060 |
| FIN Flag Count | 58 | 0.083 | 0.048 |
| Fwd IAT Max | 56 | 0.075 | 0.077 |
| Flow Packets/s | 52 | 0.059 | 0.056 |

To gain further insight into the decision-making process of the classifier, we examined individual predictions using SHAP waterfall plots. Figure 6 illustrates the contribution of each feature to a specific false negative decision. The base value $E[f(x)]$ corresponds to the expected model output across the training data (approximately 0.32 in our case), and each bar shows the impact of a single feature. Positive SHAP values push the prediction toward class 1 (malicious), while negative values push it toward class 0 (benign).

In this example, although the ground truth is *malicious*, features such as `FIN Flag Count` and `Fwd IAT Max` contribute negatively, driving the classifier to output a low probability of attack and ultimately misclassify the sample as *benign*.

We also aggregated the most influential features across all false negatives. Table 4 lists the five most frequent features that appeared in the top 10 contributors for individual misclassified samples, along with their average absolute SHAP values. While the summary plot captures overall importance across all records, this table highlights which features consistently appeared among the most influential in individual false negative cases.

This analysis confirms that even in rare failure cases, the classifier's decisions are driven by a combination of plausible feature contributions. However, in certain borderline

Table 5: Parameters used for the experts evaluation.

| System | Notes and parameters |
|---|---|
| Naive Bayes | Multinomial NB |
| Decision Tree | Maximum depth: 50, split criterion: information gain |
| Random Forest | 100 estimators, with max depth of 50 |
| Neural Network | 3 layers. Input: 128 neurons with ReLU, output: 15 neurons with softmax |

Table 6: Overall expert classifier performance comparison.

| Metric | NB | DT | RF | NN |
|---|---|---|---|---|
| Accuracy | 84.83% | 98.02% | **99.27%** | 97.02% |
| Precision | 83.12% | 99.08% | **99.45%** | 97.03% |
| Recall | 84.83% | 98.02% | **98.27%** | 96.41% |
| $F_1$-score (weighted) | 81.74% | 98.34% | **99.31%** | 96.72% |
| AUC-PR (macro) | 0.74 | 0.98 | **0.99** | 0.96 |
| MCC | 0.68 | 0.94 | **0.96** | 0.92 |
| Training time | **5.31s** | 128.27s | 636.56s | 259.32s |
| Classification time | 1.13s | **0.29s** | 15.79s | 19.57s |

cases, these contributions may align in a way that pushes the predicted score just below the classification threshold. These findings suggest potential directions for future improvement, such as threshold calibration, feature engineering, or incorporating uncertainty-aware mechanisms.

### *4.5. Experts evaluation*

This section evaluates and compares the four types of classifiers used in the MIDES ensemble on the CIC-IDS-2017 dataset. In addition to traditional metrics such as accuracy, precision, recall, and $F_1$-score, we also report the MCC and the macro-averaged AUC-PR, which provide a more robust evaluation under class imbalance and reflect the classifier's ability to generalize across diverse attack types. It is important to highlight that the performance metrics reported here refer to individual classifiers in isolation, used as part of the ensemble layer. While these results provide insights into the behavior of each model, the final detection performance of MIDES is achieved through the coordinated combination of all classifiers, the weighted voting mechanism, and the meta-classifier layer, guided by the arbiter's expertise and confidence heuristics. This integration compensates for the limitations of individual classifiers, ensuring that the system as a whole maintains robust detection capabilities across all attack classes. The effectiveness of this design is demonstrated in the global evaluation reported in the subsequent sections.

Table 6 shows the overall results obtained by the various classifiers, while Table 7 shows the $F_1$-score achieved for each class to compare them in more detail, and Table 5 summarizes the parameters used for each classifier. Decision Tree and Random Forest present very high values for all metrics: average accuracy, precision, recall and $F_1$-score are consistently in the 98-99% range. This highlights how these two classifiers are an excellent choice, as they recognize with great accuracy both attacks and any benign records that had been misclassified by the Gatekeeper. High MCC values, such as those obtained by Decision Tree and Random Forest, confirm the consistency of these classifiers in correctly labeling both majority and minority classes. Similarly, the macro AUC-PR reflects the ability to maintain good precision-recall tradeoffs across all classes, not just the most frequent ones. Once again, Random Forest and Decision Tree stand out with values above 0.98, further confirming their robustness. Despite these strong overall results, a detailed class-wise analysis remains essential. Global metrics may still mask poor performance on specific minority classes. Therefore, Table 7 reports the $F_1$-scores achieved by each classifier for every attack class.

Table 7: Comparison of the $F_1$-scores obtained by Naive Bayes (NB), Decision Tree (DT), Random Forest (RF) and Neural Network (NN).

| Classifier | NB | DT | RF | NN |
|---|---|---|---|---|
| Bot | 0.5% | 74.5% | **75.2%** | 72.3% |
| DDoS | 0.3% | 20.5% | 38.5% | **45.1%** |
| DoS GoldenEye | 13.7% | 86.5% | **88.9%** | 78.1% |
| DoS Hulk | 18.5% | 86.1% | **91.1%** | 75.3% |
| DoS Slowhttptest | 80.8% | **99.3%** | 99.0% | 91.4% |
| DoS slowloris | 26.7% | 94.8% | **97.4%** | 74.7% |
| FTP-Patator | 0.6% | **65.9%** | 58.3% | 54.4% |
| Heartbleed | 97.5% | **99.8%** | 99.7% | 95.2% |
| Infiltration | 52.5% | **99.8%** | 99.4% | 88.4% |
| PortScan | 6.7% | 38.4% | 34.2% | **42.2%** |
| SSH-Patator | 92.2% | 95.0% | **96.1%** | 87.1% |
| WA w/ Brute Force | 90.0% | **90.9%** | 84.2% | 83.2% |
| WA w/ Sql Injection | 89.2% | 89.9% | **99.4%** | 65.8% |
| WA w/ XSS | 0.3% | 58.3% | **80.0%** | 77.3% |

Indeed, global metrics hide the ability of individual classifiers to recognize specific attack classes, and, as previously mentioned, this is precisely the major weakness of single systems compared to ensembles and meta-classifiers in general. Other systems such as Naive Bayes, which performs significantly worse when evaluating only overall performance, can still be useful, since they are potentially "expert" classifiers in recognizing critical classes, which pose a challenge to better global classifiers.

This is clearly highlighted by Table 7, which shows the $F_1$-score obtained by the four systems for each attack class. DDoS, PortScan, and FTP-Patator classes are difficult to recognize for both Decision Tree and Random Forest, resulting in distinctly low $F_1$-score values. However, Random Forest obtains a good score for Web Attack w/ XSS (80%), which is difficult to classify for the Decision Tree (58.3%). As for the other classes, the two classifiers behave similarly, with Random Forest generally having a slight advantage over Decision Tree.

Naive Bayes was the fastest classifier for training, followed by Decision Tree, Neural Network, and finally Random Forest, which was the slowest. For classification, De-

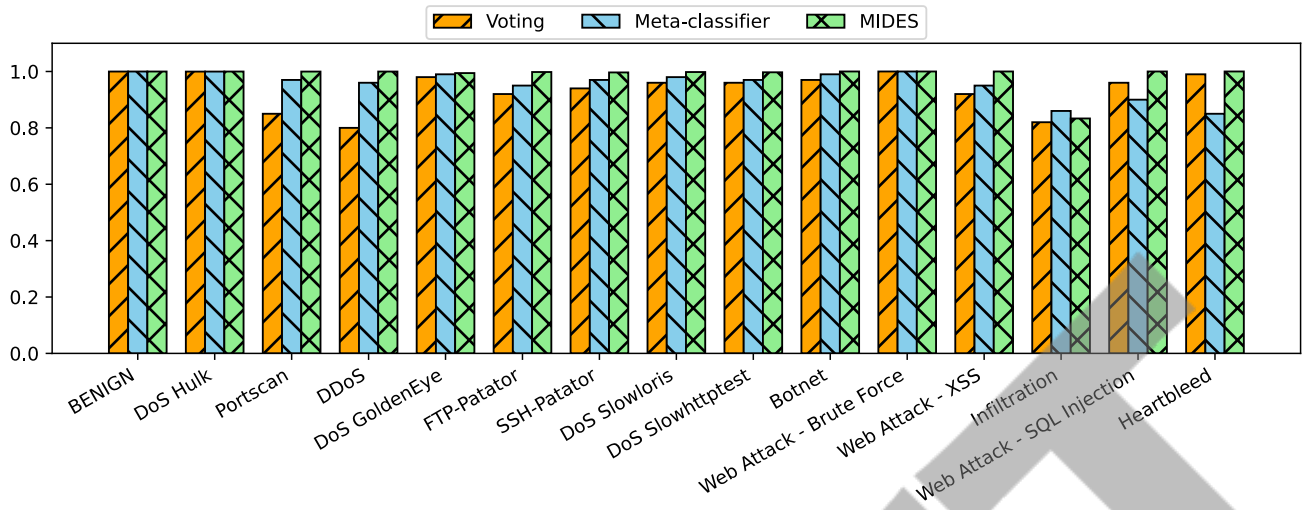



Figure 7: $F_1$-scores obtained by the voting module, the meta-classifier, and the complete MIDES system for each attack class.

cision Tree was the fastest, followed by Naive Bayes, with Random Forest and Neural Network being the slowest, in that order. Although Naive Bayes is very fast, however, its performance leaves much to be desired, with low $F_1$-score values. Nevertheless, some classes are well recognized even by this classifier, namely DoS Slowhttptest, Heartbleed, SSH-Patator, Web Attack w/ Brute Force and Web Attack w/ Sql Injection. Notably, Naive Bayes obtains a higher score than Random Forest for the Web Attack w/ Brute Force attack, and its ability to classify Heartbleed and Web Attack w/ Sql Injection is noteworthy as well, considering that these two specific classes are challenging for more complex systems, as will be shown in the next section.

Finally, the Neural Network remains somewhat in the middle, never excelling, but it is still the slowest system of the four as regards classification time. Nevertheless, the combined use of all four of these classifiers, with their respective strengths and weaknesses, contributes to the performance of MIDES as a whole, as will be shown in the next section. These results are also confirmed by the macro-averaged $F_1$-score, which equally weights all attack classes. According to this metric, the scores are 79%, 82%, 41% and 74% for Decision Tree, Random Forest, Naive Bayes and Neural Network, respectively.

### *4.6. Impact of the Voting, Meta-Classifier, and Arbiter modules on system performance*

This section compares the performance of the Voting module, the Meta-Classifier, and the complete MIDES system, which integrates the two modules through the Arbiter to achieve optimal decision-making. Figure 7 shows the $F_1$-scores achieved by each system for all attack classes.

The Voting module generally yields lower performance than the Meta-Classifier. However, it achieves competitive results in some minority classes. For example, in Heartbleed and Web Attack – SQL Injection, the Voting module obtains $F_1$-scores of 0.99 and 0.96, outperforming the Meta-Classifier, which scores 0.85 and 0.90 in the same classes. These results indicate that the Voting module can be advantageous when dealing with classes that are underrepresented in the training set.

Conversely, the Voting module struggles with several attack classes, such as PortScan, DDoS, and Infiltration, with $F_1$-scores of 0.85, 0.80, and 0.82, respectively. Overall, a system based solely on expert voting is unsatisfactory due to its inconsistent performance across classes.

The Meta-Classifier achieves higher and more stable results across most classes, reaching $F_1$-scores above 0.95 for the majority of attacks. However, it still shows weaknesses for certain minority classes: Heartbleed (0.85), Web Attack – SQL Injection (0.90), and Infiltration (0.86). These results confirm that the Meta-Classifier, while generally strong, has limited generalization capability in classes with very few records, even after over-sampling.

Ideally, the self-adaptive agent of MIDES should be able to estimate and predict the performance of the Voting and Meta-Classification modules, and leverage the Expertise and Confidence metrics that were presented in Section 3.3.3 to select the best module to use at any given time. To validate the performance of the self-adaptive agent, Figure 7 also shows the $F_1$-score obtained by MIDES with $\gamma = 0.5$, to give equal importance to the static and dynamic terms of Equation 23.

By combining the two modules, MIDES benefits from the strengths of both approaches. The Arbiter dynamically leverages the Expertise and Confidence metrics discussed in Section 3.3.3 to select the most suitable module for each decision. As a result, MIDES achieves consistently high $F_1$-scores in nearly all classes. In particular, MIDES markedly improves the results in classes where the meta-classifier struggles, reaching almost perfect scores in Heartbleed and Web Attack – SQL Injection and obtaining higher values in FTP-Patator. In the only case where the

$F_1$-score slightly decreases compared to the meta-classifier (i.e., Infiltration), the drop is negligible.

The overall improvements are reflected in the macro-averaged $F_1$-scores of 0.938, 0.956, and 0.988 for the voting module, meta-classifier, and MIDES, respectively. Similarly, the macro-averaged AUC-PR values are 0.788, 0.806, and 0.979, while the MCC values are 0.969, 0.978, and 0.999. These metrics confirm that MIDES not only achieves superior accuracy but also maintains strong precision-recall trade-offs and balanced predictions, correctly identifying both benign and attack traffic.

Overall, these results demonstrate that MIDES significantly outperforms its individual submodules. The self-adaptive agent effectively estimates the expected performance of the two decision modules and selects the most reliable one, thereby increasing both robustness and accuracy, especially in minority classes where individual modules alone may fail.

### *4.7. Comparisons with state-of-the-art systems*

This section presents the comparative analysis of MIDES against two recent state-of-the-art IDSs: Leader Class and Confidence Decision Ensemble (LCCDE) and NEFI-IDS. LCCDE, proposed in [84, 83], is a framework that selects the ML model with the best performance among three algorithms (XGBoost, LightGBM, and CatBoost) for each class or type of attack. The class-leading models, with their prediction confidence scores, are then used to make accurate decisions regarding the detection of different types of cyber-attacks. The second system considered in the presented comparison, NEFI-IDS, was proposed in [64] and uses several algorithms that are ranked by F1-score for their ability to detect specific attacks. Among them, LightGBM and HBGB were particularly effective, demonstrating high detection rates and low prediction latency. Both models show strong classification capabilities against different categories of attacks.

To ensure a robust and fair comparison, the systems were tested on two different datasets (CIC-IDS2017 and CSE-CIC-IDS2018). Evaluating the three systems on multiple datasets is useful to verify their effectiveness against a variety of attacks, ensuring robustness in different environments. In addition, this approach provides a more accurate assessment of the detection rate and mitigates the risk of biased results due to dataset-specific characteristics.

The comparative analysis was performed by running each IDS under identical conditions and using the same evaluation metrics: accuracy, precision, recall, $F_1$-score, macro-averaged AUC-PR, MCC, and execution time. For each dataset, the execution of multiple runs ensures statistical significance, by averaging the results obtained.

On the CIC-IDS2017 dataset, as presented in Table 8, MIDES achieved a macro F1-score of 98.78%, outperforming LCCDE (97.46%) and NEFI-IDS (88.87%). However, while MIDES excelled in this metric, LCCDE achieved a slightly higher weighted F1-score compared to MIDES. This discrepancy can be attributed to LCCDE's ability to handle imbalanced classes effectively, which is particularly relevant in cases where certain attack types are less prevalent. The macro-averaged AUC-PR values confirm MIDES's superiority in balancing precision and recall under class imbalance (0.98 for MIDES vs. 0.96 and 0.83 for LCCDE and NEFI-IDS, respectively). Similarly, the MCC values are extremely high for all systems due to their near-perfect classification of benign traffic, but MIDES still achieves the highest value, confirming its consistently correct predictions across both majority and minority classes.

Table 8: Performance comparison of the three systems on the CIC-IDS2017 dataset.

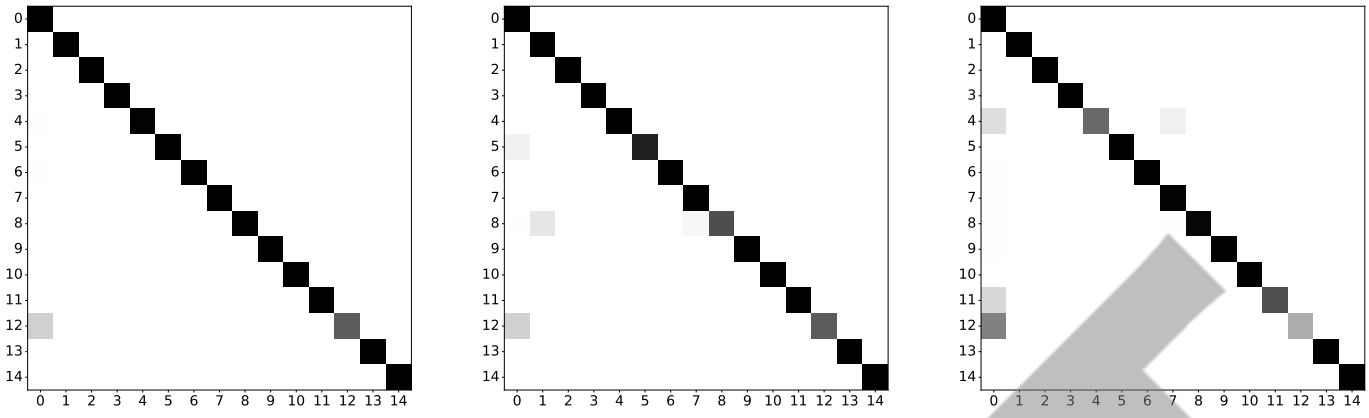
| Metric | MIDES | LCCDE | NEFI-IDS |
|---|---|---|---|
| Accuracy | **99.98%** | 99.95% | 99.85% |
| Precision | **99.98%** | 99.81% | 89.28% |
| Recall | **97.90%** | 95.70% | 91.88% |
| $F_1$-score (macro) | **98.78%** | 97.46% | 88.87% |
| $F_1$-score (weighted) | 99.98% | **99.99%** | 99.84% |
| AUC-PR (macro) | **0.9787** | 0.9551 | 0.8313 |
| MCC | **0.9995** | 0.9988 | 0.9963 |
| Classification time | **109s** | 2691s | 691s |

Table 9: Performance comparison of the three systems on the CSE-CIC-IDS2018 dataset.

| Metric | MIDES | LCCDE | NEFI-IDS |
|---|---|---|---|
| Accuracy | **99.97%** | 99.30% | 92.83% |
| Precision | **99.39%** | 95.73% | 94.90% |
| Recall | 95.69% | 94.77% | **97.10%** |
| $F_1$-score (macro) | **97.27%** | 94.85% | 95.09% |
| $F_1$-score (weighted) | **99.97%** | 99.32% | 94.10% |
| AUC-PR (macro) | **0.9533** | 0.9115 | 0.9211 |
| MCC | **0.9995** | 0.9877 | 0.8910 |
| Classification time | **130s** | 2215s | 479s |

Analyzing other metrics, MIDES also led in accuracy (99.98%) and precision (99.98%), demonstrating a robust detection capability and a low rate of false positives. The execution time of 109 seconds highlights MIDES's efficiency, especially when compared to LCCDE's 2691 seconds and NEFI-IDS's 691 seconds. This indicates that the filtering performed by the Gatekeeper lead to significant improvements in execution times, as will be shown in the next section.

The confusion matrices shown in Fig. 8 reveal that MIDES performs well across all attack classes, with robust detection capabilities overall. However, the class "Infiltration" poses some challenges, as it is difficult to recognize for all three systems. This attack is frequently misclassified as benign traffic, which can be attributed to both the low number of records associated with the class in the dataset and the nature of the attack itself, which often resembles innocuous traffic. For the "Infiltration" class, MIDES achieves an F1-score of 83.3%, while LCCDE scores 82.1%,



(a) MIDES (CIC-IDS2017) (b) LCDDE (CIC-IDS2017) (c) NEFI (CIC-IDS2017)

Figure 8: Confusion Matrix for the three compared systems in CIC-IDS2017. Class legend: 0 - Benign, 1 - DoS Hulk, 2 - Portscan, 3 - DDoS, 4 - DoS GoldenEye, 5 - FTP-Patator, 6 - SSH-Patator, 7 - DoS Slowloris, 8 - DoS Slowhttptest, 9 - Botnet, 10 - Web Attack - Brute Force, 11 - Web Attack - XSS, 12 - Infiltration, 13 - Web Attack - SQL Injection, 14 - Heartbleed.

and NEFI-IDS obtains 52.5%. This highlights MIDES's relative effectiveness in identifying this challenging class, although there remains room for improvement.

In addition to "Infiltration", LCCDE also shows some uncertainty in recognizing the "FTP - Patator" and "DoS Slowhttptest" classes, as evidenced by the confusion matrix results. Conversely, NEFI-IDS struggles more with the "DoS GoldenEye" and "Web Attack - XSS" classes. These observations highlight the need for improvements in the detection of less common and easily confused attack types, which are critical to improving the overall effectiveness of Intrusion Detection Systems.

In the CSE-CIC-IDS2018 dataset (Table 9), MIDES again demonstrated superior performance with a macro F1-score of 97.27%, outpacing both LCCDE (94.85%) and NEFI-IDS (95.09%). The weighted F1-score for MIDES was also higher (99.97%) than that of NEFI-IDS (94.10%), suggesting a balanced performance across the attack classes. The advantage of MIDES is again confirmed by its macro AUC-PR of 0.95, compared to 0.91 for LCCDE and 0.92 for NEFI-IDS. MCC values are extremely high for both MIDES and LCCDE, reflecting their near-perfect overall accuracy. NEFI-IDS achieves a lower MCC of 0.89, highlighting its comparatively higher rate of misclassifications. MIDES still obtains the highest value, reinforcing its robustness and balanced handling of both false positives and false negatives.

In terms of execution time, MIDES maintained its advantage, completing its classification in 130 seconds compared to 2215 seconds for LCCDE and 479 seconds for NEFI-IDS.

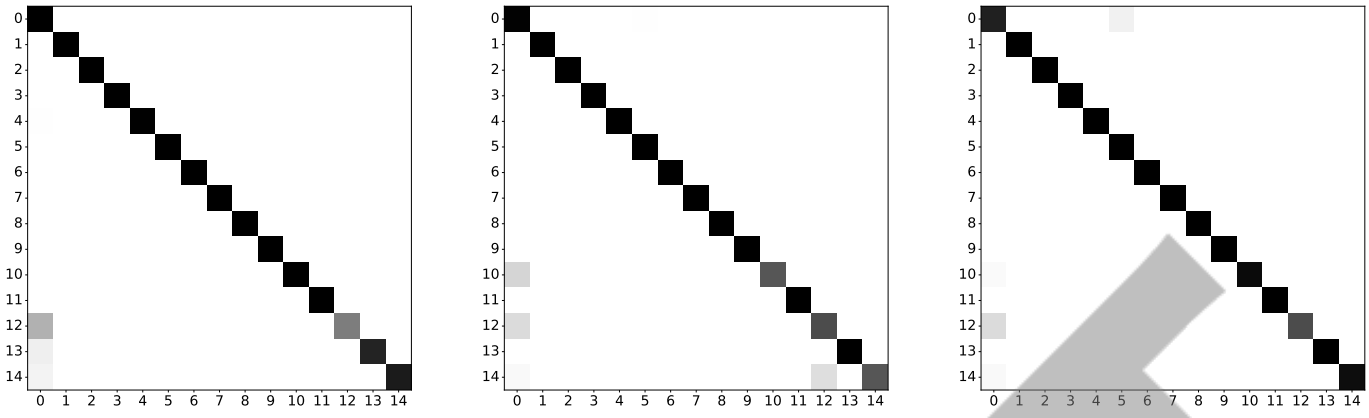
The detailed analysis of the confusion matrices in Fig. 9 shows how MIDES demonstrates strong performance across most attack classes in the CSE-CIC-IDS2018 dataset, with difficulties primarily arising in the class "Infiltration - Dropbox Download" (which presents a common challenge for all three systems), and to a lesser extent with two classes that have few records in the dataset, specifically "Web Attack - SQL" and "Infiltration - Communication Victim Attacker". The latter class also poses challenges for LCCDE and NEFI-IDS, as both systems tend to misclassify it as different attack types.

It is also worth noticing that the class "Web Attack - Brute Force" is successfully identified by MIDES but causes some errors for the other two systems. Additionally, NEFI-IDS incorrectly classifies a non-negligible percentage of benign instances, achieving an F1-score of 94.1% for this category. In contrast, MIDES obtains an F1-score of 99.9% for this class, due to the effectiveness of the Gatekeeper, while LCCDE follows with an F1-score of 99.4%.

This comparative analysis highlights the better performance of MIDES in terms of both detection accuracy and computational efficiency. MIDES's architecture, which combines multiple classifiers and uses a meta-classifier for final decision making, proves to be more effective than the ensemble approaches used by LCCDE and NEFI-IDS. The initial filtering provided by the Gatekeeper significantly reduces the computational burden on subsequent layers, allowing MIDES to maintain high performance even under high traffic volumes. In addition, MIDES' ability to integrate new classifiers as needed ensures that it remains adaptable to emerging threats, providing a future-proof solution for intrusion detection.

### *4.8. Impact of the Gatekeeper*

To measure the impact of the Gatekeeper on system performance, MIDES is evaluated both with and without the Gatekeeper, with the expectation that including the Gatekeeper would significantly improve execution time. The logic behind this expectation is that by pre-filtering benign traffic, the Gatekeeper reduces the computational burden on the ensemble layer and the meta-classifier. As a result, these layers can operate more efficiently, focusing on analyzing traffic that is more likely to be malicious.



(a) MIDES (CSE-CIC-IDS2018) (b) LCDDE (CSE-CIC-IDS2018) (c) NEFI (CSE-CIC-IDS2018)

Figure 9: Confusion Matrix for the three compared systems in CSE-CIC-IDS2018. Class legend: 0 - Benign, 1 - DoS Hulk, 2 - DDoS-HOIC, 3 - DDoS-LOIC-UDP, 4 - DoS GoldenEye, 5 - Infiltration - NMAP Portscan, 6 - SSH-BruteForce, 7 - DoS Slowloris, 8 - DDoS-LOIC-HTTP, 9 - Botnet Ares, 10 - Web Attack - Brute Force, 11 - Web Attack - XSS, 12 - Infiltration - Dropbox Download, 13 - Web Attack - SQL, 14 - Infiltration - Communication Victim Attacker.

Each configuration (with and without Gatekeeper) was tested under identical conditions. The experimental setup involved running each test 1000 times to ensure statistical significance.

Table 10 compares the execution times of MIDES with and without the Gatekeeper for the CIC-IDS2017 dataset. The results show a significant decrease in classification time when the Gatekeeper is used, highlighting its effectiveness in reducing the system's computational overhead. Specifically, the classification time with the Gatekeeper is just 109 seconds, compared to 432 seconds without it, representing a 75% reduction. Table 11 presents similar results for the CSE-CIC-IDS2018 dataset, where the classification time with the Gatekeeper is reduced to 130 seconds, compared to 363 seconds without it, demonstrating a 64.2% improvement in efficiency. Meanwhile, the overheads during training are small, with times increasing by 8% and 7% for CIC-IDS2017 and CSE-CIC-IDS2018, respectively.

Tables 10 and 11 also show that the inclusion of the Gatekeeper does not negatively affect the system's classification performance. MIDES achieves very similar results across all evaluation metrics, including $F_1$-score (both macro and weighted), macro-averaged AUC-PR, and MCC, with only small variations depending on the dataset.

For the CIC-IDS2017 dataset, the F1-score (macro) is actually higher when the Gatekeeper is active. This improvement is due to the fact that the Gatekeeper filters out only those samples that are confidently classified as benign. As a result, the downstream ensemble and meta-classifier layers receive a more focused subset of the traffic, which improves classification on less frequent attack classes. Since the macro-average F1-score gives equal weight to all classes, better performance on minority classes has a strong positive impact on this metric. This is also reflected in the AUC-PR and MCC scores,

Table 10: System performance with and without Gatekeeper (GK), CIC-IDS2017 dataset.

| | **With GK** | **Without GK** |
|---|---|---|
| $F_1$-score (macro) | **98.78%** | 94.38% |
| $F_1$-score (weighted) | 99.98% | **99.99%** |
| AUC-PR (macro) | **0.98** | 0.96 |
| MCC | **0.99** | 0.97 |
| Training time | 1501s (1.08x) | **1384s (1.0x)** |
| Classification time | **109s (0.25x)** | 432s (1.0x) |

which improve from 0.96 to 0.98 and from 0.97 to 0.99, respectively, indicating better precision-recall trade-offs and more balanced binary decision performance across classes.

On the other hand, the F1-score (weighted) slightly decreases, because this metric is dominated by the benign class, which represents the majority of the traffic. Filtering out some of these benign samples removes "easy" true positives, slightly lowering the weighted average.

In the CIC-IDS2018 dataset, both macro and weighted F1-scores are marginally lower with the Gatekeeper. This behavior is likely due to the higher heterogeneity of this dataset, which includes attack classes that are harder to distinguish from benign traffic. In such cases, even a conservative Gatekeeper may in very rare circumstances misclassify an attack as benign, leading to a minor drop in recall for those classes. This effect is visible in both macro AUC-PR, which slightly decreases when the Gatekeeper is active (0.95 vs. 0.97), although performance remains consistently high. Nevertheless, in both datasets, the classification performance remains high, while the inclusion of the Gatekeeper brings a substantial reduction in classification time. This efficiency gain makes MIDES more suitable for real-time deployment, justifying the use of the Gatekeeper despite the minor fluctuations in F1-score.

These results confirm that the Gatekeeper significantly

Table 11: System performance with and without Gatekeeper (GK), CSE-CIC-IDS2018 dataset.

| | **With GK** | **Without GK** |
|---|---|---|
| $F_1$-score (macro) | 97.27% | **98.84%** |
| $F_1$-score (weighted) | 99.97% | **99.99%** |
| AUC-PR (macro) | 0.95 | **0.97** |
| MCC | **0.99** | **0.99** |
| Training time | 1285s (1.07x) | **1197s (1.0x)** |
| Classification time | **130s (0.36x)** | 363s (1.0x) |

Table 12: Performance comparison of the three systems trained on CIC-IDS2017 and tested on CIC-IDS2018.

| **Metric** | **MIDES** | **LCCDE** | **NEFI-IDS** |
|---|---|---|---|
| Accuracy | **70.6%** | 66.4% | 61.9% |
| $F_1$-score (macro) | **56.2%** | 52.8% | 48.7% |
| $F_1$-score (weighted) | **63.3%** | 60.1% | 55.4% |
| AUC-PR | **0.618** | 0.554 | 0.492 |
| MCC | **0.347** | 0.296 | 0.242 |

reduces the overall execution time, allowing the other layers to focus on a smaller subset of the traffic, thus improving the overall efficiency and accuracy of the system. This is most noticeable in scenarios with high traffic volumes and diverse attack types, demonstrating the critical role of the Gatekeeper in ensuring real-time performance and high reliability.

### *4.9. Robustness under distribution shift and Limitations*

To assess the robustness of MIDES under distribution shifts and previously unseen attacks, we conducted cross-dataset experiments by training the system on CIC-IDS2017 and testing it on CIC-IDS2018. Because the two datasets differ in both feature distributions and attack categories, the classification task was binarized into benign vs. attack to ensure label compatibility.

Table 12 reports the results in comparison with LCCDE and NEFI-IDS. MIDES consistently outperformed the other two approaches across all metrics. As expected, the recall decreased significantly due to the presence of attack types not seen during training and the change in traffic patterns. Nonetheless, MIDES preserved the best $F_1$-scores among the three systems, demonstrating its greater ability to generalize to new traffic distributions.

To further simulate a scenario where the network suddenly encounters traffic with different characteristics, we tested MIDES on the concatenation of CIC-IDS2017 and CIC-IDS2018 after training it on CIC-IDS2017. This setup mimics a *sudden concept drift* event, where the system is first exposed to familiar traffic and then to a distribution containing novel attacks and altered statistical properties.

As shown in Table 13, MIDES maintained high accuracy, while achieving reasonable $F_1$-score and MCC despite the abrupt introduction of previously unseen traffic patterns. The performance drop is consistent with the lack of online adaptation in the current implementation, highlighting a potential direction for future work involving incremental or online learning to handle evolving traffic conditions.

Table 13: Performance of MIDES under a sudden concept drift scenario (training on CIC-IDS2017, testing on CIC-IDS2017+2018).

| **Metric** | **MIDES** |
|---|---|
| Accuracy | 85.2% |
| $F_1$-score (macro) | 79.2% |
| $F_1$-score (weighted) | 83.6% |
| AUC-PR | 0.810 |
| MCC | 0.649 |

These results suggest that although MIDES is more robust than the compared systems to distribution shifts, its performance can still degrade when facing sudden concept drift. Integrating online learning or adaptive model update strategies could further enhance the ability of MIDES to handle evolving network traffic and unseen attacks in real-world deployments.

## 5. Conclusions and Future Work

This paper described MIDES, a multi-layer Intrusion Detection System that employs an ensemble of multiple classifiers, including binary, multi-class, and meta-classifiers, to improve its performance.

MIDES exploits a binary classifier to quickly filter all incoming data. Since malicious traffic typically only makes up a small portion of total traffic, this initial filtering process significantly reduces the number of events that need to be analyzed by the rest of the system, resulting in faster overall response time. Events that are considered suspicious are further evaluated through an ensemble of classifiers. To determine the final classification, the system includes a self-adaptive agent, called the Arbiter, which dynamically chooses between a weighted voting module and a meta-classifier. This decision is based on a heuristic score that combines a static measure of each classifier's expertise with a dynamic estimation of prediction confidence. MIDES further enhances its performance by exploiting a set of meta-features to construct a meta-dataset, enabling a second-level classification stage via stacked generalization. This architecture improves the ability to distinguish between similar attack patterns and adapt to varying input characteristics without retraining the base models.

The system was thoroughly evaluated on two widely used IDS datasets through a series of experiments designed to validate its individual components and overall architecture. The results confirm that MIDES achieves high classification accuracy across 14 different attack types and outperforms other state-of-the-art solutions in both predictive performance and classification speed. Additional experiments included an ablation study, which demonstrated the benefit of the adaptive Arbiter module, and a cross-dataset evaluation to measure resilience under distribution shift.

These stress tests highlighted the strengths of MIDES under stable conditions, as well as its current limitations in dynamic or evolving environments.

Several directions will be explored as future work to enhance the flexibility, scalability, and adaptability of the system. Firstly, online learning mechanisms can be integrated to allow MIDES to update its models incrementally as new data becomes available, thereby enabling continuous learning without full retraining. In parallel, we will investigate concept drift detection and adaptation techniques, aiming to maintain classification accuracy as the statistical properties of network traffic evolve over time.

Another important research direction involves the deployment of MIDES in resource-constrained environments such as IoT or edge computing platforms. This will require rethinking certain components of the ensemble to ensure computational and memory efficiency. In this regard, we plan to explore lightweight CNN architectures to reduce inference time and resource usage. To support distributed detection in decentralized scenarios, we will also evaluate the use of federated learning. This would allow individual MIDES instances deployed on different nodes to collaborate in updating global models without sharing raw data, thus preserving privacy while improving threat detection coverage across heterogeneous networks.

Finally, the MIDES architecture could be extended with unsupervised and semi-supervised components to better handle zero-day attacks and unknown traffic behaviors, where labeled data is scarce or unavailable. In particular, we plan to investigate the integration of outlier detection or unsupervised anomaly detection mechanisms as a backup for the Gatekeeper, providing an additional safeguard for borderline cases without significantly increasing computational overhead. Future work will also include robustness assessments against adversarial manipulation techniques, evaluating how MIDES reacts to crafted inputs designed to evade detection, and integrating mitigation strategies where needed. These research directions aim to further evolve MIDES into a scalable, resilient, and adaptive IDS capable of operating effectively in dynamic and heterogeneous network environments.

## Acknowledgments

This work was partially supported by the ADELE project, within the project SERICS (PE00000014), under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU.

## References

[1] Abbasi, J.S., Bashir, F., Qureshi, K.N., Najam ul Islam, M., Jeon, G., 2021. Deep learning-based feature extraction and optimizing pattern matching for intrusion detection using finite state machine. Computers & Electrical Engineering 92, 107094. doi:https://doi.org/10.1016/j.compeleceng.2021.107094.

[2] Aburomman, A.A., Reaz, M.B.I., 2017. A survey of intrusion detection systems based on ensemble and hybrid classifiers. Computers & Security 65, 135–152. doi:https://doi.org/10.1016/j.cose.2016.11.004.

[3] Addula, S.R., Mamodiya, U., Jiang, W., Almaiah, M.A., 2025. Generative AI-enhanced intrusion detection framework for secure healthcare networks in manets. SHIFRA 2025, 62–68.

[4] Agate, V., Concone, F., De Paola, A., Ferraro, P., Lo Re, G., Morana, M., 2024. Adaptive ensemble learning for intrusion detection systems, in: CEUR Workshop Proceedings.

[5] Agate, V., D'Anna, F.M., De Paola, A., Ferraro, P., Lo Re, G., Morana, M., 2022. A behavior-based intrusion detection system using ensemble learning techniques., in: CEUR Workshop Proceedings, 6th Italian Conference on Cybersecurity, ITASEC 2022, pp. 207–218.

[6] Al-Daweri, M.S., Abdullah, S., Ariffin, K.A.Z., 2021. A homogeneous ensemble based dynamic artificial neural network for solving the intrusion detection problem. International Journal of Critical Infrastructure Protection 34, 100449.

[7] Alfrhan, A.A., Alhusain, R.H., Khan, R.U., 2020. Smote: Class imbalance problem in intrusion detection system, in: 2020 International Conference on Computing and Information Technology (ICCIT-1441), IEEE. pp. 1–5.

[8] Alhowaide, A., Alsmadi, I., Tang, J., 2020. Pca, random-forest and pearson correlation for dimensionality reduction in iot ids, in: 2020 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS), pp. 1–6. doi:10.1109/IEMTRONICS51293.2020.9216388.

[9] Alhowaide, A., Alsmadi, I., Tang, J., 2021. Ensemble detection model for iot ids. Internet of Things 16, 100435. doi:https://doi.org/10.1016/j.iot.2021.100435.

[10] Aljuhani, A., Alamri, A., Jolfaei, A., 2025. Lightweight fuzzy-driven intrusion detection for consumer life-tech applications. IEEE Transactions on Consumer Electronics , 1–1doi:10.1109/TCE.2025.3569886.

[11] Alsajri, A., Steiti, A., 2024. Intrusion detection system based on machine learning algorithms:(SVM and genetic algorithm). Babylonian Journal of Machine Learning 2024, 15–29.

[12] Amor, N.B., Benferhat, S., Elouedi, Z., 2004. Naive bayes vs decision trees in intrusion detection systems,

in: Proceedings of the 2004 ACM symposium on Applied computing, pp. 420–424.

[13] Aung, Y.Y., Myat Min, M., 2018. Hybrid intrusion detection system using k-means and k-nearest neighbors algorithms, in: 2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS), pp. 34–38. doi:10.1109/ICIS.2018.8466537.

[14] Bhuyan, M.H., Bhattacharyya, D.K., Kalita, J.K., 2014. Network anomaly detection: Methods, systems and tools. IEEE Communications Surveys Tutorials 16, 303–336. doi:10.1109/SURV.2013.052213.00046.

[15] Buczak, A.L., Guven, E., 2016. A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Communications Surveys Tutorials 18, 1153–1176. doi:10.1109/COMST.2015.2494502.

[16] Chang, H., Feng, J., Duan, C., 2020. Hadiot: A hierarchical anomaly detection framework for iot. IEEE Access 8, 154530–154539.

[17] Choraś, M., Pawlicki, M., 2021. Intrusion detection approach based on optimised artificial neural network. Neurocomputing 452, 705–715.

[18] Chui, K.T., Gupta, B.B., Chaurasia, P., Arya, V., Almomani, A., Alhalabi, W., 2023. Three-stage data generation algorithm for multiclass network intrusion detection with highly imbalanced dataset. International Journal of Intelligent Networks 4, 202–210. doi:https://doi.org/10.1016/j.ijin.2023.08.001.

[19] Creech, G., Hu, J., 2014. A semantic approach to host-based intrusion detection systems using contiguousand discontiguous system call patterns. IEEE Transactions on Computers 63, 807–819. doi:10.1109/TC.2013.13.

[20] Džeroski, S., Ženko, B., 2004. Is combining classifiers with stacking better than selecting the best one? Machine learning 54, 255–273.

[21] El-Khatib, K., 2009. Impact of feature reduction on the efficiency of wireless intrusion detection systems. IEEE TRANSACTIONS on parallel and distributed systems 21, 1143–1149.

[22] Eskandari, M., Janjua, Z.H., Vecchio, M., Antonelli, F., 2020. Passban ids: An intelligent anomaly-based intrusion detection system for iot edge devices. IEEE Internet of Things Journal 7, 6882–6897. doi:10.1109/JIOT.2020.2970501.

[23] Gaddam, S.R., Phoha, V.V., Balagani, K.S., 2007. K-means+ id3: A novel method for supervised anomaly detection by cascading k-means clustering and id3 decision tree learning methods. IEEE transactions on knowledge and data engineering 19, 345–354.

[24] García-Teodoro, P., Díaz-Verdejo, J., Maciá-Fernández, G., Vázquez, E., 2009. Anomaly-based network intrusion detection: Techniques, systems and challenges. Computers & Security 28, 18–28. doi:https://doi.org/10.1016/j.cose.2008.08.003.

[25] Généreux, S.J.J., Lai, A.K.H., Fowles, C.O., Roberge, V.R., Vigeant, G.P.M., Paquet, J.R., 2020. Maidens: Mil-std-1553 anomaly-based intrusion detection system using time-based histogram comparison. IEEE Transactions on Aerospace and Electronic Systems 56, 276–284. doi:10.1109/TAES.2019.2914519.

[26] Ghadermazi, J., Hore, S., Shah, A., Bastian, N.D., 2025. Gtae-ids: Graph transformer-based autoencoder framework for real-time network intrusion detection. IEEE Transactions on Information Forensics and Security 20, 4026–4041. doi:10.1109/TIFS.2025.3557741.

[27] Hong, J., Liu, C.C., Govindarasu, M., 2014. Integrated anomaly detection for cyber security of the substations. IEEE Transactions on Smart Grid 5, 1643–1653. doi:10.1109/TSG.2013.2294473.

[28] Huang, Y., Ma, M., 2025. Aill-ids: An automatic incremental lifetime learning intrusion detection system for vehicular ad hoc networks. IEEE Transactions on Intelligent Transportation Systems 26, 2669–2678. doi:10.1109/TITS.2024.3510584.

[29] Iglesias Pérez, S., Moral-Rubio, S., Criado, R., 2021. A new approach to combine multiplex networks and time series attributes: Building intrusion detection systems (ids) in cybersecurity. Chaos, Solitons & Fractals 150, 111143. doi:https://doi.org/10.1016/j.chaos.2021.111143.

[30] Ioulianou, P., Vasilakis, V., Moscholios, I., Logothetis, M., 2018. A signature-based intrusion detection system for the internet of things. Information and Communication Technology Form .

[31] Jang-Jaccard, J., Nepal, S., 2014. A survey of emerging threats in cybersecurity. Journal of Computer and System Sciences 80, 973–993. doi:https://doi.org/10.1016/j.jcss.2014.02.005. special Issue on Dependable and Secure Computing.

[32] Kabir, E., Hu, J., Wang, H., Zhuo, G., 2018. A novel statistical technique for intrusion detection systems. Future Generation Computer Systems 79, 303–318.

[33] Khraisat, A., Gondal, I., Vamplew, P., Kamruzzaman, J., 2019. Survey of intrusion detection systems: techniques, datasets and challenges. Cybersecurity 2, 1–22.

[34] Koc, L., Mazzuchi, T.A., Sarkani, S., 2012. A network intrusion detection system based on a hidden naïve bayes multiclass classifier. Expert Systems with Applications 39, 13492–13500.

[35] Kotpalliwar, M.V., Wajgi, R., 2015. Classification of attacks using support vector machine (svm) on kddcup'99 ids database, in: 2015 Fifth International Conference on Communication Systems and Network Technologies, IEEE. pp. 987–990.

[36] Krishna Kishore, P., Ramamoorthy, S., Rajavarman, V., 2023. Artp: Anomaly based real time prevention of distributed denial of service attacks on the web using machine learning approach. International Journal of Intelligent Networks 4, 38–45. doi:https://doi.org/10.1016/j.ijin.2022.12.001.

[37] Kumar, I., Mohd, N., Bhatt, C., Sharma, S.K., 2020. Development of ids using supervised machine learning, in: Soft computing: Theories and applications. Springer, pp. 565–577.

[38] Li, X., Zhu, M., Yang, L.T., Xu, M., Ma, Z., Zhong, C., Li, H., Xiang, Y., 2021. Sustainable ensemble learning driving intrusion detection model. IEEE Transactions on Dependable and Secure Computing 18, 1591–1604. doi:10.1109/TDSC.2021.3066202.

[39] Lin, M., Yang, K., Yu, Z., Shi, Y., Chen, C.P., 2023. Hybrid ensemble broad learning system for network intrusion detection. IEEE Transactions on Industrial Informatics .

[40] Lin, Y.D., Lu, Y.H., Hwang, R.H., Lai, Y.C., Sudyana, D., Lee, W.B., 2025. Evolving ml-based intrusion detection: Cyber threat intelligence for dynamic model updates. IEEE Transactions on Machine Learning in Communications and Networking 3, 605–622. doi:10.1109/TMLCN.2025.3564587.

[41] Liu, L., Engelen, G., Lynar, T., Essam, D., Joosen, W., 2022. Error prevalence in nids datasets: A case study on cic-ids-2017 and cse-cic-ids-2018, in: 2022 IEEE Conference on Communications and Network Security (CNS), IEEE. pp. 254–262.

[42] Lu, Q., An, K., Li, J., Wang, J., 2025. Network intrusion detection for modern smart grids based on adaptive online incremental learning. IEEE Transactions on Smart Grid 16, 2541–2553. doi:10.1109/TSG.2025.3535949.

[43] Lundberg, S.M., Lee, S.I., 2017. A unified approach to interpreting model predictions, in: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.), Advances in Neural Information Processing Systems 30. Curran Associates, Inc., pp. 4765–4774.

[44] Masdari, M., Khezri, H., 2020. A survey and taxonomy of the fuzzy signature-based intrusion detection systems. Applied Soft Computing 92, 106301. doi:https://doi.org/10.1016/j.asoc.2020.106301.

[45] Masoodi, H.J.K.A., 2024. Evaluating the effectiveness of machine learning-based intrusion detection in multi-cloud environments. Babylonian Journal of Internet of Things 2024, 94–105.

[46] McHugh, J., 2000. Testing intrusion detection systems: A critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. ACM Trans. Inf. Syst. Secur. 3, 262–294. doi:10.1145/382912.382923.

[47] Muda, Z., Yassin, W., Sulaiman, M.N., Udzir, N.I., 2011a. Intrusion detection based on k-means clustering and naïve bayes classification, in: 2011 7th International Conference on Information Technology in Asia, pp. 1–6. doi:10.1109/CITA.2011.5999520.

[48] Muda, Z., Yassin, W., Sulaiman, M.N., Udzir, N.I., 2011b. Intrusion detection based on k-means clustering and oner classification, in: 2011 7th International Conference on Information Assurance and Security (IAS), pp. 192–197. doi:10.1109/ISIAS.2011.6122818.

[49] Mukherjee, S., Sharma, N., 2012. Intrusion detection using naive bayes classifier with feature reduction. Procedia Technology 4, 119–128.

[50] Obaid, A.J., Alghurabi, K.A., Albermany, S.A., Sharma, S., 2021. Improving extreme learning machine accuracy utilizing genetic algorithm for intrusion detection purposes, in: Research in Intelligent and Computing in Engineering. Springer, pp. 171–177.

[51] Otoum, Y., Nayak, A., 2021. As-ids: Anomaly and signature based ids for the internet of things. Journal of Network and Systems Management 29, 1–26.

[52] Pajouh, H.H., Javidan, R., Khayami, R., Dehghantanha, A., Choo, K.K.R., 2019. A two-layer dimension reduction and two-tier classification model for anomaly-based intrusion detection in iot backbone networks. IEEE Transactions on Emerging Topics in Computing 7, 314–323. doi:10.1109/TETC.2016.2633228.

[53] Pang, G., Shen, C., Cao, L., Hengel, A.V.D., 2021. Deep learning for anomaly detection: A review. ACM computing surveys (CSUR) 54, 1–38.

[54] Pathak, P., Chauhan, E., Rathi, S., Kosti, S., 2018. Hmm-based ids for attack detection and prevention in manet, in: Information and Communication Technology for Sustainable Development. Springer, pp. 413–421.

[55] Pradeep, K.J., Shukla, P.K., 2025. Designing a novel network anomaly detection framework using multi-serial stacked network with optimal feature selection procedures over ddos attacks. International Journal of Intelligent Networks 6, 1–13. doi:https://doi.org/10.1016/j.ijin.2024.11.001.

[56] R, R., S, H., Nagarajan, S.M., Devarajan, G.G., Omar, M., Bashir, A.K., 2024. Threat detection and mitigation for tactile internet driven consumer iot-healthcare system. IEEE Transactions on Consumer Electronics 70, 4249–4257. doi:10.1109/TCE.2024.3370193.

[57] Rajora, K., et al., 2023. Reviews research on applying machine learning techniques to reduce false positives for network intrusion detection systems. Babylonian Journal of Machine Learning 2023, 26–30.

[58] Rieger, P., Chilese, M., Mohamed, R., Miettinen, M., Fereidooni, H., Sadeghi, A.R., 2023. ARGUS: Context-Based detection of stealthy IoT infiltration attacks, in: 32nd USENIX Security Symposium (USENIX Security 23), pp. 4301–4318.

[59] Ring, M., Wunderlich, S., Scheuring, D., Landes, D., Hotho, A., 2019. A survey of network-based intrusion detection data sets. Computers & Security 86, 147–167. doi:https://doi.org/10.1016/j.cose.2019.06.005.

[60] R.M., S.P., Maddikunta, P.K.R., M., P., Koppu, S., Gadekallu, T.R., Chowdhary, C.L., Alazab, M., 2020. An effective feature engineering for dnn using hybrid pca-gwo for intrusion detection in iomt architecture. Computer Communications 160, 139–149. doi:https://doi.org/10.1016/j.comcom.2020.05.048.

[61] Roshan, K., Zafar, A., 2024. Ensemble adaptive online machine learning in data stream: a case study in cyber intrusion detection system. International Journal of Information Technology 16, 5099–5112.

[62] Saiyed, M.F., Al-Anbagi, I., 2024. Deep ensemble learning with pruning for ddos attack detection in iot networks. IEEE Transactions on Machine Learning in Communications and Networking .

[63] Sayem, I.M., Sayed, M.I., Saha, S., Haque, A., 2024. Enids: A deep learning-based ensemble framework for network intrusion detection systems. IEEE Transactions on Network and Service Management .

[64] Seth, S., Chahal, K.K., Singh, G., 2021. A novel ensemble framework for an intelligent intrusion detection system. IEEE Access 9, 138451–138467. doi:10.1109/ACCESS.2021.3116219.

[65] Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A., 2018. Toward generating a new intrusion detection dataset and intrusion traffic characterization. ICISSp 1, 108–116.

[66] Shiravi, A., Shiravi, H., Tavallaee, M., Ghorbani, A.A., 2012. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. Computers & Security 31, 357–374. doi:https://doi.org/10.1016/j.cose.2011.12.012.

[67] Tama, B.A., Comuzzi, M., Rhee, K.H., 2019. Tse-ids: A two-stage classifier ensemble for intelligent anomaly-based intrusion detection system. IEEE Access 7, 94497–94507. doi:10.1109/ACCESS.2019.2928048.

[68] Tan, J., Huang, L., Xia, Z., Gu, K., Hao, W., Long, K., Zeng, L., 2025. Dati-ids: Domain adaptation and time-series imaging-based intrusion detection system for connected autonomous vehicles. IEEE Transactions on Intelligent Transportation Systems , 1–18doi:10.1109/TITS.2025.3587750.

[69] Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A., 2009. A detailed analysis of the kdd cup 99 data set, in: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, pp. 1–6. doi:10.1109/CISDA.2009.5356528.

[70] Tavallaee, M., Stakhanova, N., Ghorbani, A.A., 2010. Toward credible evaluation of anomaly-based intrusion-detection methods. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 40, 516–524. doi:10.1109/TSMCC.2010.2048428.

[71] Ten, C.W., Hong, J., Liu, C.C., 2011. Anomaly detection for cybersecurity of the substations. IEEE Transactions on Smart Grid 2, 865–873. doi:10.1109/TSG.2011.2159406.

[72] Ting, K.M., Witten, I.H., 1999. Issues in stacked generalization. Journal of artificial intelligence research 10, 271–289.

[73] Tseng, C.Y., Balasubramanyam, P., Ko, C., Limprasittiporn, R., Rowe, J., Levitt, K., 2003. A specification-based intrusion detection system for aodv, in: Proceedings of the 1st ACM Workshop on Security of Ad Hoc and Sensor Networks, Association for Computing Machinery, New York, NY, USA. pp. 125–134. doi:10.1145/986858.986876.

[74] Ullah, I., Khalil, I., Bai, X., Garg, S., Kaddoum, G., Shamim, M., 2025. An ensemble-based hybrid model for the detection of attacks in the internet of vehicular things. IEEE Transactions on Intelligent Transportation Systems , 1–14doi:10.1109/TITS.2025.3547999.

[75] Van Rijsbergen, C.J., 1979. Information retrieval. 2nd. newton, ma.

[76] Viinikka, J., Debar, H., Mé, L., Lehikoinen, A., Tarvainen, M., 2009. Processing intrusion detection alert aggregates with time series modeling. Information Fusion 10, 312–324.

[77] Wei, L., Yang, J., Jin, H., Cui, J., Li, J., He, D., 2025. Sustainable learning-based intrusion detection system for vanets. IEEE Transactions on Intelligent Transportation Systems , 1–12doi:10.1109/TITS.2025.3562226.

[78] Wolpert, D.H., 1992. Stacked generalization. Neural networks 5, 241–259.

[79] Wolpert, D.H., Macready, W.G., 1997. No free lunch theorems for optimization. IEEE transactions on evolutionary computation 1, 67–82.

[80] Wu, M., Zheng, Y., Wong, D.S.H., Wang, Y., Hu, X., 2025. Tracer: Attack-aware divide-and-conquer transformer for intrusion detection in industrial internet of things. IEEE Transactions on Industrial Informatics , 1–11doi:10.1109/TII.2025.3547050.

[81] Xiao, Y., Xing, C., Zhang, T., Zhao, Z., 2019. An intrusion detection model based on feature reduction and convolutional neural networks. IEEE Access 7, 42210–42219. doi:10.1109/ACCESS.2019.2904620.

[82] Xue, Y., Pan, J., Geng, Y., Yang, Z., Liu, M., Deng, R., 2024. Real-time intrusion detection based on decision fusion in industrial control systems. IEEE Transactions on Industrial Cyber-Physical Systems 2, 143–153. doi:10.1109/TICPS.2024.3406505.

[83] Yang, L., Moubayed, A., Shami, A., 2021. Mth-ids: A multitiered hybrid intrusion detection system for internet of vehicles. IEEE Internet of Things Journal 9, 616–632.

[84] Yang, L., Shami, A., Stevens, G., De Rusett, S., 2022. Lccde: a decision-based ensemble framework for intrusion detection in the internet of vehicles, in: GLOBECOM 2022-2022 IEEE Global Communications Conference, IEEE. pp. 3545–3550.

[85] Yang, X., Tong, F., Jiang, F., Cheng, G., 2025. A lightweight and dynamic open-set intrusion detection for industrial internet of things. IEEE Transactions on Information Forensics and Security 20, 2930–2943. doi:10.1109/TIFS.2025.3546849.

[86] Yu, X., Lu, Y., Jiang, F., Hu, Q., Du, J., Gong, D., 2025. A cross-domain intrusion detection method based on nonlinear augmented explicit features. IEEE Transactions on Network and Service Management 22, 187–197. doi:10.1109/TNSM.2024.3444909.

[87] Zavrak, S., İskefiyeli, M., 2020. Anomaly-based intrusion detection from network flow features using variational autoencoder. IEEE Access 8, 108346–108358. doi:10.1109/ACCESS.2020.3001350.

[88] Zha, C., Wang, Z., Fan, Y., Bai, B., Zhang, Y., Shi, S., Zhang, R., 2025. A-nids: Adaptive network intrusion detection system based on clustering and stacked ctgan. IEEE Transactions on Information Forensics and Security 20, 3204–3219. doi:10.1109/TIFS.2025.3551643.

[89] Zhong, Y., Wang, Z., Shi, X., Yang, J., Li, K., 2024. Rfg-helad: A robust fine-grained network traffic anomaly detection model based on heterogeneous ensemble learning. IEEE Transactions on Information Forensics and Security 19, 5895–5910. doi:10.1109/TIFS.2024.3402439.

[90] Zhou, Y., Cheng, G., Jiang, S., Dai, M., 2020. Building an efficient intrusion detection system based on feature selection and ensemble classifier. Computer Networks 174, 107247. doi:https://doi.org/10.1016/j.comnet.2020.107247.