


A Framework for Drift Detection and Adaptation in AI-driven Anomaly and Threat Detection Systems

Antonio Lara-Gutierrez ^{*} Carmen Fernandez-Gago [†]

Jose A. Onieva [‡]

*Network, Information and Computer Security (NICS) Lab
University of Málaga, Málaga, Spain*

September 2, 2025

Abstract

The dynamic and evolving nature of cybersecurity threats presents significant challenges to anomaly and threat detection systems, particularly those that rely on Artificial Intelligence (AI) as their detection engine. A key limitation of current AI models is their inability to adapt to concept drift, feature drift, and adversarial attacks, which degrade performance over time. Although these phenomena arise from different underlying processes, they all share the effect of misaligning the operational data with the model’s training data. This study introduces the Hybrid Drift Detection and Adaptation Framework (HDDAF), which is a multi-layered AI system that is specifically designed to mitigate concept drift, feature drift, and adversarial attacks in cybersecurity. By framing all three challenges, HDDAF provides a unified approach that detects and responds to both natural evolution and malicious manipulation within a single adaptive pipeline. HDDAF integrates Hoeffding drift detection, feature selection, adversarial training, and incremental learning, allowing it to dynamically adapt through a Mixed-Drift Handling Module, which balances fine-tuning and full retraining. On the CIC-IDS2017 dataset, HDDAF achieves a macro F1 score

above 99% and in tests on related datasets, it consistently adapts to data shifts with minimal retraining. An ablation study confirms that each module contributes to overall robustness, and real-time simulations demonstrate its ability to process high-velocity streams with stable latency and resource use. HDDAF’s hybrid design delivers both high accuracy and scalable performance for real-world cybersecurity applications.

Keywords: Artificial Intelligence; Concept Drift; Feature Drift; Adversarial Attacks

1 Introduction

Intrusion Detection Systems (IDS) emerged from the increasing demand for proactive security measures to monitor system-level activities and detect unauthorised or malicious behaviour within a system or network. Anderson’s foundational work in the 1970s introduced the concept of monitoring user actions for suspicious activities [4], thus laying the groundwork for contemporary IDS practices. Building on this foundation, Denning proposed a formal model in the 1980s that emphasised real-time anomaly detection and audit data analysis as critical security strategies [9].

As cyberattacks have become increasingly complex and sophisticated, there is a growing need for more dynamic IDSs capable of handling diverse, large-

^{*}alara@uma.es

[†]mcgago@uma.es

[‡]onieva@uma.es

scale, and real-time scenarios. Early IDS solutions primarily relied on static rule-based engines and statistical analysis, which quickly proved inadequate for scalability and for detecting novel threats. The integration of Artificial Intelligence (AI) techniques, including Machine Learning (ML) and Deep Learning (DL), has introduced a paradigm shift in the capabilities of IDS.

ML models are valued for their interpretability; however, they struggle with scalability and high-dimensional data. In contrast, DL models excel at capturing complex temporal and spatial patterns but are computationally demanding. Hybrid models that combine ML and DL techniques have emerged as promising alternatives, offering a balance between performance and computational efficiency.

Despite these advancements, existing AI models face critical challenges when deployed in real-world cybersecurity environments. Concept drift [17], where data distributions evolve, and feature drift [5], where the importance of the feature changes and degrades model performance, may compromise future model outputs. In addition, adversarial attacks exploit vulnerabilities in AI systems, which reduces their robustness and reliability. Developing adaptive methods capable of detecting and mitigating drift while maintaining system performance can address these issues [29].

In cybersecurity, these drift phenomena have concrete implications. Concept drift occurs when the behaviour of attacks evolves over time, like a denial-of-service attack, which may change in frequency, packet structure, or targeted ports, making previously learned detection rules obsolete. Feature drift, on the other hand, arises when the distribution of normal network behaviour changes, such as shifts in user activity patterns, protocol usage, or device configurations, which can lead to increased false positives or missed detections. These dynamics are common in real-world networks, where both attackers and legitimate users continuously adapt. Thus, handling drift is not only a theoretical necessity, but a practical requirement for maintaining accurate and reliable threat detection in operational settings.

Although concept drift, feature drift, and adversarial examples are the result of different underlying

causes, they share a unifying effect: each introduces a distribution shift between training data and operational data encountered during deployment [18]. Concept drift alters the conditional distribution $P(y | x)$, leading to outdated decision boundaries; feature drift shifts the marginal distribution $P(x)$, causing input characteristics to move outside the original feature space; and adversarial examples represent deliberate perturbations designed to push inputs away from the clean training manifold. By framing these phenomena as instances of distributional misalignment, the paper aims to propose a framework which successfully applies a cohesive detection and adaptation strategy to handle both natural evolution in traffic patterns and maliciously crafted inputs under a single unified approach.

Many existing frameworks fail to provide a comprehensive approach that dynamically handles both concept and feature drift while maintaining system resilience and long-term adaptability.

To address these challenges, we introduce the Hybrid Drift Detection and Adaptation Framework (HDDAF), which mainly consists of an enhanced framework designed for AI-driven anomaly and threat detection systems. HDDAF integrates a multilayered hybrid AI architecture with active adaptation modules to maintain detection performance in dynamic threat landscapes. The primary contributions of this work are as follows.

1. A mixed drift handling module that dynamically identifies and responds to concept and feature drifts.
2. An adaptive generative model leverages GANs to pre-train the model for future predictive drift.
3. Robustness against adversarial attacks through data poisoning using adversarial training techniques.
4. The experimental evaluation demonstrates that the proposed method effectively handles real-time data, exhibits strong generalization between datasets, and benefits from each architectural component, as confirmed by an ablation study.

This article is structured as follows. Section 2 reviews previous works that address anomaly detection, drift management, and adaptive AI systems. Section 3 details the proposed methodology, including its architectural design and components. Section 4 presents the results of the experimental evaluation and discusses the insights extracted from the development. Finally, Section 6 concludes the paper, highlighting future research directions to further enhance AI-driven anomaly detection systems.

2 Related Work

The challenge of managing evolving data distributions in AI-driven anomaly and threat detection has become a critical focus in recent years. Numerous frameworks and methodologies have been introduced to enhance the adaptability, robustness, and overall performance of such systems. Table 1 provides a comparative overview of various studies, highlighting the various areas they address and their contributions to the field.

A key point in this field is the integration of explainability and dynamic adaptability into drift detection frameworks. Zhang et al. (2024) [31] pioneered the HCDD-SHAP method, a multilayer drift detection approach that combines model explainability with precise drift identification. The framework sequentially detects drift through error rate variations, validates it using SHAP values to assess feature contributions, and identifies the drift onset point with a Kneedle algorithm. This layered design enhances interpretability and precision, making it particularly effective in contexts requiring transparency.

Complementing Zhang et al.’s work, the study presented by Mulimani et al. (2024) [19] introduced an online drift detection and adaptation framework leveraging the Light Gradient Boosting Machine (LGBM). Their approach integrates incremental learning, ensemble methods, and a sliding window technique to detect and adapt to concept drift in real-time data streams. The experiments demonstrated superior accuracy and adaptability compared to traditional models. While Zhang et al. emphasised explainability, Mulimani et al.’s framework excelled

in dynamic adaptation, making it better suited for real-time applications.

Aguiar and Cano (2023) [3] proposed a meta-learning framework that dynamically selects drift detectors based on statistical and temporal features. By tailoring detection mechanisms to the characteristics of evolving data, the proposed system highlights the importance of flexibility when addressing diverse scenarios.

To address feature drift explicitly, Noori et al. (2023) [21] introduced the Dynamic Feature-Aware GP Ensemble (DFA-GPE), which integrates a Variable Length Multi-Objective Particle Swarm Optimisation (VLMO-PSO) technique. This framework dynamically manages feature importance, achieving memory efficiency and high accuracy in high-dimensional datasets like HIKARI ¹ and TON-IoT 2020 ². Unlike Zhang et al. and Mulimani et al., who focused on concept drift and adaptation, Noori et al. excelled in environments where feature importance evolves over time, which is a critical factor in handling high-dimensional data.

Expanding the concept to hybrid model designs, Wisanwanichthan et al. (2021) [27] combined Naive Bayes and SVM classifiers into a double-layered approach. Validated on the NSL-KDD ³ dataset, this system effectively handled various attack patterns, demonstrating the value of targeted designs for specific applications.

Incorporating optimisation techniques, ElDahshan et al. (2022) [10] used meta-heuristic algorithms to fine-tune hierarchical intrusion detection systems. The proposed method not only improved detection rates but also demonstrated the potential of hybrid optimisation methods to address complex data distributions.

Deep learning-based methodologies also show promise. Halbouni et al. (2022) [12] combined CNN and LSTM architectures to capture spatial and temporal patterns in network traffic. Their

¹HIKARI dataset available at <https://zenodo.org/record/5199540>

²TON-IoT datasets available at <https://research.unsw.edu.au/projects/toniot-datasets>

³NSL-KDD dataset available at <https://web.archive.org/web/20150205070216/http://nsl.cs.unb.ca/NSL-KDD/>

Reference	Approach	Concept Drift	Feature Drift	Feature Engineering	Adversarial Attacks	Dataset(s) Used
Zhang et al. (2024) [31]	HCDD-SHAP (Explainability-based multilayer drift detection)	✓	✗	✓	✗	Synthetic, Real-World
Mulimani et al. (2024) [19]	Online LightGBM with Sliding Window and Incremental Learning	✓	✗	✓	✗	Electricity, Spam, MixedAbrupt
Aguiar and Cano (2023) [3]	Meta-Learning for Concept Drift Detection	✓	✗	✓	✗	10 Real-World + 18 Synthetic
Noori et al. (2023) [21]	Dynamic Feature Aware GP Ensemble (DFA-GPE)	✓	✓	✓	✗	HIKARI 2021, TON_IoT 2020
Wisnwanichthan et al. (2021) [27]	Double-Layered Hybrid Approach (Naive Bayes + SVM)	✓	✓	✓	✗	NSL-KDD
ElDahshan et al. (2022) [10]	Meta-Heuristic Optimisation-Based Hierarchical IDS	✓	✓	✓	✗	UNSW-NB15, CIDS2017
Halbouni et al. (2022) [12]	CNN-LSTM Hybrid Deep Neural Network	✓	✓	✓	✗	CIC-IDS2017, UNSW-NB15, WSN-DS
Zhao et al. (2024) [32]	GAN-Enhanced NIDS for Imbalanced Traffic	✓	✓	✓	✓	CIC-IDS2017
Abdelaty et al. (2022) [1]	GADoT: GAN-based Adversarial Training for Robust DDoS Detection	✗	✗	✓	✓	CIC-IDS2017, UNSW-NB15
Yuan et al. (2023) [30]	AE Detection via Local Intrinsic Dimensionality	✗	✗	✓	✓	CIC-IDS2017, Custom
Sheikhi et al. (2022) [25]	PSOGWO-Optimised BP Neural Network	✓	✓	✓	✗	NSL-KDD
Harwahyu et al. (2024) [13]	Three-layer hybrid learning (LSTM + RF)	✓	✓	✓	✗	CSE-CIC-IDS2018
Turukmane et al. (2024) [26]	M-MultiSVM with Advanced Feature Selection	✓	✓	✓	✗	CSE-CIC-IDS2018, UNSW-NB15
Our Proposal	Hybrid Drift Detection and Adaptation Framework (HDDAF)	✓	✓	✓	✓	CICIDS2017 and adversarial data

Table 1: Comparison of Related Works on Drift Detection and Adaptation Frameworks.

hybrid model achieved notable success in detecting anomalies across datasets such as CIC-IDS2017 ⁴ and UNSW-NB15 ⁵, underscoring the efficacy of neural networks in cybersecurity contexts.

Switching to adversarial attacks and adversarial samples generation and to address the challenge of imbalanced attack classes, Zhao et al. (2024) [32] proposed the GAN-Enhanced NIDS, which generates realistic synthetic network traffic for rare attack categories using Vanilla GAN, WGAN, and CTGAN, thereby significantly improving detection accuracy on CIC-IDS2017 by mitigating class imbalance through

high-fidelity samples.

Focusing on adversarial robustness, GADoT was introduced by Abdelaty et al. (2022) [1], a GAN-based adversarial training framework that injects crafted DDoS samples into the training pipeline, forcing the model to learn resilient decision boundaries and reducing undetected malicious flows from over 60% to below 2% on both CIC-IDS2017 and UNSW-NB15.

To enhance defense against crafted perturbations, Yuan et al. (2023) [30] proposed a hybrid IDS architecture incorporating a Local Intrinsic Dimensionality-based adversarial example detector alongside both deep learning and traditional machine learning classifiers, dynamically filtering out adversarial noise before classification and improving

⁴CIC-IDS2017 dataset available at <https://www.unb.ca/cic/datasets/ids-2017.html>

⁵UNSW-NB15 dataset available at <https://research.unsw.edu.au/projects/unsw-nb15-dataset>

robustness on CIC-IDS2017.

A comparative analysis of these studies revealed common threads in the pursuit of enhanced drift detection and adaptation frameworks. Many authors have advocated continuous model updates and dynamic mechanisms for drift detection, reflecting the practical constraints of real-world applications. The proposed framework synthesises these insights with the aim of extending the capabilities of existing models, thereby offering a comprehensive solution to the multifaceted problem of drift in dynamic environments.

3 Hybrid Drift Detection and Adaptation Framework (HDDAF)

In this paper, we have presented a scalable and adaptive framework for anomaly and threat detection called the Hybrid Drift Detection and Adaptation Framework (HDDAF), is presented. The proposed framework is designed to address the evolving challenges posed by concept drift, feature drift, and adversarial attacks in AI-driven systems. The system integrates multi-layered hybrid AI approaches with advanced preprocessing, dynamic drift adaptation mechanisms, and adversarial training to ensure long-term resilience and maintainability over time. The proposed approach is presented in Figure 1.

3.1 Dataset

The dataset that represents the real traffic data in the proposed framework is CIC-IDS2017 [24], developed by the Canadian Institute for Cybersecurity using a university network with different subnets and mixed real traffic. The dataset includes 14 types of labelled individual attacks, which we have grouped in 8 families, namely Bot, Brute Force, DDoS, DoS, Heartbleed, Infiltration, Port Scan, and Web Attack. It is widely used for benchmarking intrusion detection systems due to its comprehensive coverage of network behaviours and threats. The dataset comprises 85 features representing flow-level statistics, with each

row representing a data flow.

However, recent studies [15, 16] have reported the presence of annotation errors, duplicated flows, and inconsistencies in the CIC-IDS2017 and CIC-IDS2018 datasets, which may introduce biases in model evaluation and lead to over-optimistic performance estimates. In our implementation, we have applied corrective preprocessing steps to mitigate some of these issues. Specifically, we removed duplicated flow entries and addressed the known port scan mislabelling problem to improve data reliability for training and evaluation.

An initial Exploratory Data Analysis (EDA) was conducted to better understand the characteristics of the dataset. Figure 2 shows the distribution of labels and the proper distribution of attacks found, categorised into 8 different types of attacks. Notably, class imbalance was observed, with benign traffic dominating over attack instances, highlighting the need for balancing techniques during the model training phase.

Table 2 highlights the features with the highest positive correlation against the target variable, providing valuable insights into the underlying patterns that distinguish different attack behaviours. Features such as “Bwd Packet Length Std” and “Bwd Packet Length Max”, with correlations of 0.33 and 0.32, respectively, indicate a strong relationship with the occurrence of specific attack types. These high correlations suggest that changes in backward packet length metrics are particularly influential in identifying attacks, possibly due to their sensitivity to data flow pattern anomalies.

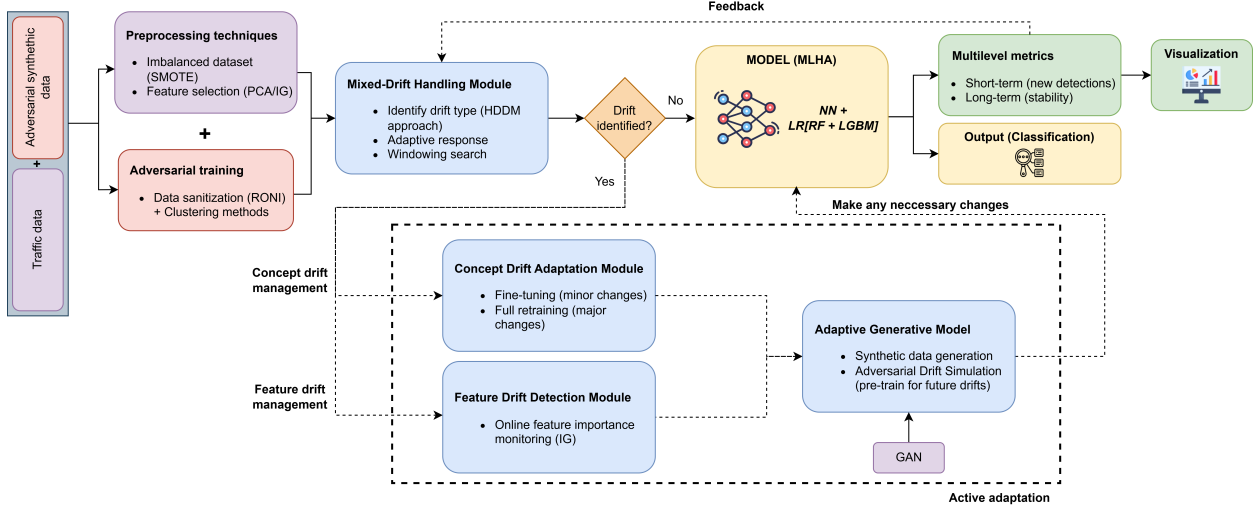


Figure 1: Hybrid Drift Detection and Adaptation Framework (HDDAF).

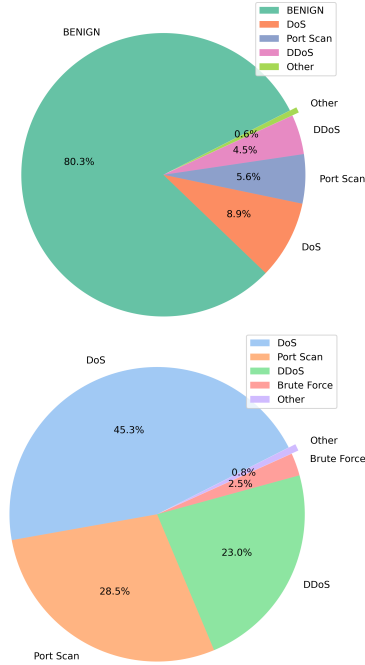


Figure 2: Attack Distribution in CIC-IDS2017: Overall Traffic (left) and Attack Types Breakdown (right).

Feature	Correlation	Feature	Correlation
Bwd Packet Length Std	0.33	Idle Max	0.29
Bwd Packet Length Max	0.32	Flow IAT Max	0.28
Bwd Packet Length Mean	0.31	Fwd IAT Max	0.28
Fwd IAT Std	0.31	Idle Mean	0.28
Avg Bwd Segment Size	0.31	Idle Min	0.28
Packet Length Std	0.30	PSH Flag Count	0.27
Max Packet Length	0.29	Packet Length Mean	0.25
Packet Length Variance	0.29	Average Packet Size	0.25

Table 2: Features with Positive Correlation with Attack Number.

3.2 Preprocessing Techniques

Different preprocessing techniques were applied to the raw dataset. For example, feature selection is implemented after minor adjustments, such as null filling, duplicate removal, or numeric filtering, to enhance the computational load and reduce complexity. The following two approaches were evaluated:

- **Principal Component Analysis (PCA) [2]:** A dimensionality reduction method that transforms features into a smaller set of uncorrelated components (called PCX). In this case, 98.58% of the information was retained, leaving the

dataset with only 36 features and the target variable. However, interpretability is no longer possible as the variables are now transformed into uncorrelated components (PCs), which leaves us with no information about model decisions, crucial for debugging, compliance, and making informed decisions in critical decisions or audits.

- **Information Gain (IG) [11]:** Measures the reduction in entropy relative to the target variable when a single feature is used for classification. Essentially, it quantifies how well a feature separated the classes in a dataset. Then, we can select some features that we want to keep for further analysis, arranged by their IG results. To manage the number of features that will be selected, we conducted a test to evaluate the classification performance of the datasets by including a successive number of features, ordered by their IG value. Figure 3 shows a moving average of the accuracy and F1 scores obtained depending on the number of selected features. The optimal value that will be chosen for future experiments will be 32.

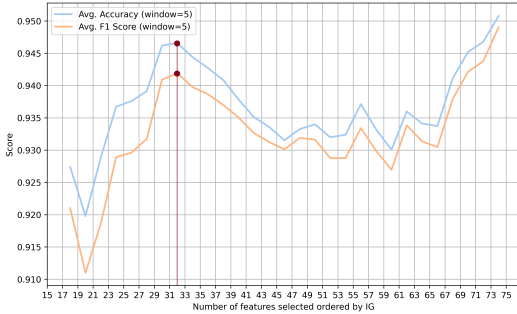


Figure 3: Rolling Average of Accuracy and F1 Score vs. Number of Selected Features Ordered by IG values.

Given the previously identified class imbalance, the synthetic minority oversampling technique (SMOTE) was applied [8]. SMOTE can generate synthetic examples for minority classes to balance the dataset. In

this case, as seen in Figure 2, these classes correspond to the rarest attacks.

When comparing SMOTE to other sampling approaches, such as random undersampling of the majority (benign) class, we observed that undersampling discards valuable benign samples, affecting model generalization, and often degrades classifier performance by increasing false positives and lowering macro F1, as also stated in [6]. In contrast, SMOTE preserves all original benign instances while synthesizing informative minority-class examples. This leads to higher macro F1 without sacrificing majority-class diversity and avoiding repetition. Consequently, SMOTE was chosen to maintain the full benign distribution and generate new attack samples, rather than merely enlarging the dataset [8, 14].

To strengthen the model’s resilience against adversarial attacks, a hybrid adversarial training technique was developed, combining the next techniques:

- **Data Sanitisation with RONI (Reject on Negative Impact) [20]:** RONI takes each sample, removes it from the dataset, and compares the validation accuracy obtained with a baseline reference value. If the accuracy was improved, the sample was flagged as ‘Rejected’. While this filtering helps filter out harmful or misleading data, it also leads to the removal of rare but important samples, which reduces model generalizability. Additionally, the method’s reliance on baseline accuracy makes it sensitive to noise and potential biases in the reference model. The overuse of RONI could result in a sanitised but overly selective dataset, limiting the model’s adaptability to diverse real-world scenarios.
- **Clustering methods:** Used to identify groups of data points, where outliers are separated into separate clusters.
- **Hybrid Adversarial Training Technique:** This technique identifies possible outliers using clustering methods (specifically, with the KMeans [28] algorithm) and then applies RONI individually to suspicious data points. The proposed approach reduces computational costs by limiting RONI evaluations to high-risk samples

rather than the entire dataset. The proposed method also mitigates the issue of removing rare but valuable samples by preserving samples that align with recognised data clusters. Furthermore, by combining clustering with RONI, the proposed method becomes less sensitive to noise and potential biases in baseline accuracy, thereby ensuring a more balanced and adaptable model. Figure 4 shows a graphical representation of how this technique works and how it affects the data, which will be analysed by the model in future steps.

To assess the impact of each sample on model performance, we employ a lightweight Random Forest classifier as the reference model. Rather than setting a fixed rejection threshold Θ , we adopt a confidence-based statistical approach. Let A_i denote the accuracy after adding the i -th sample, and A_{ref} be the accuracy with clean data. A sample is rejected if:

$$A_i < A_{\text{ref}} - \Theta, \quad \text{where} \quad \Theta = |\mu_{\Delta A} - Z \cdot \sigma_{\Delta A}|$$

Here, $\mu_{\Delta A}$ is the mean change in accuracy when clean samples are added, $\sigma_{\Delta A}$ is the corresponding standard deviation, and Z is the Z-score representing the confidence level (we use $Z = 1.96$ for a 95% confidence level). This formulation helps ensure that only statistically significant degradations in model accuracy result in sample rejection, providing robustness while preserving potentially informative samples.

When trying to obtain the maximum precision, RONI excels because it manages to obtain a low number of false positives (high precision), but the execution time can be extensive. However, clustering methods often identify many false positives, resulting in low precision but high sensitivity. To achieve balance in all these aspects, the proposed hybrid technique works seamlessly in all these aspects, as shown in the comparison plotted on Figure 5.

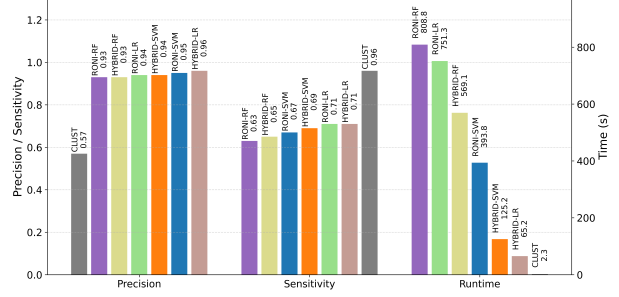


Figure 5: Precision, Sensitivity, and Runtime Comparison of Selected Techniques for Data Sanitisation.

3.3 Hybrid Model

The HDDAF employs a multilayered hybrid AI architecture to maintain robustness and high detection accuracy under evolving conditions. In addition, we implement measures to retrain the model and maintain a high accuracy value.

The hybrid model, depicted in Figure 6, combines a neural network (NN) comprising two hidden layers with an ensemble of Random Forest (RF) and LightGBM (LGBM), finalised with a Logistic Regression (LR) meta-classifier. The NN serves as the initial detection layer, extracting features, and performing binary classification to distinguish between benign traffic and potential anomalies. The deep architecture of the proposed method enables the detection of complex patterns, while dropout regularisation helps reduce overfitting. In comparison with other approaches, binary classification using an NN demonstrated high precision and low sensitivity, making it highly suitable for this task. Then, the ensemble layer combines RF and LGBM, where RF provides feature importance and variance reduction, and LGBM offers gradient boosting with efficient handling of large datasets and imbalanced traffic patterns.

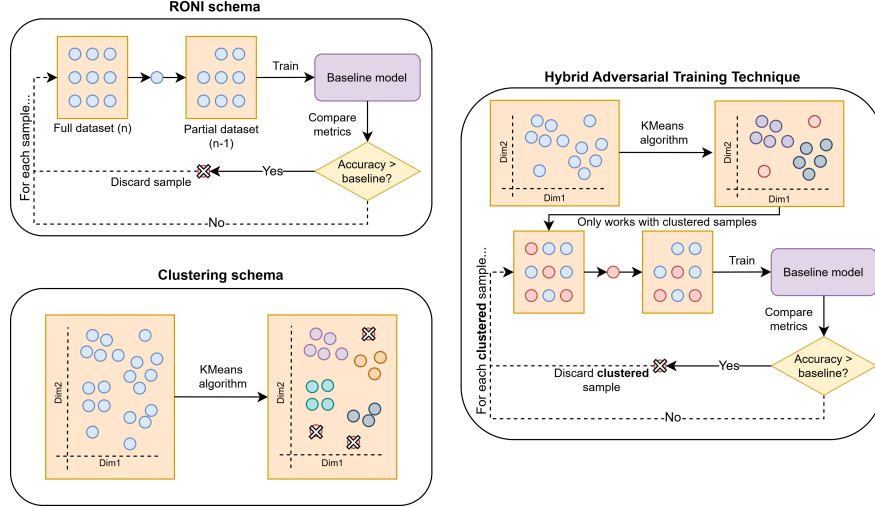


Figure 4: Graphical Representation of RONI, Clustering and Hybrid Adversarial Training Technique Working Principles.

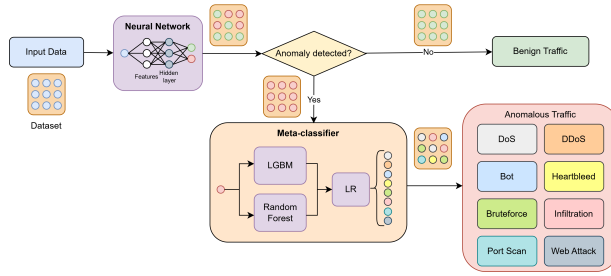


Figure 6: Representation of the Hybrid Model Proposed for Classification.

The final decision layer employs a logistic regression meta-classifier to aggregate predictions from both RF and LGBM, combining their outputs in a weighted manner to enhance overall accuracy. The use of a Logistic Regression meta-classifier for the final aggregation ensures a unified and optimised output by combining predictions in a weighted manner. The specifications of the implementations are as detailed in Table 4. The selected hyperparameters are obtained from a cross-validated grid-search over a parameter grid for each classifier used.

The choice of Logistic Regression was driven by its low complexity, interpretability, and strong generalization capability, making it suitable as a lightweight aggregator for ensemble outputs. However, to validate this design decision, we evaluated stacking performance using alternative meta-learners including Random Forest, XGBoost, MLP, and a soft-voting ensemble. Table 3 presents a comparative analysis of these models based on key performance and practical criteria. Although MLP and Voting showed similar performance to LR in terms of accuracy and F1-score, Logistic Regression was ultimately chosen due to its faster training time, higher interpretability, and lower risk of overfitting in real-time environments.

Meta-Classifer	Acc.	F1	Train (s)	Infer (s)	Interp./Overfit
LR	0.9989	0.9989	63.74	0.0200	High / Low
RF	0.9900	0.9901	81.48	0.0189	Med / Med
XGBoost	0.9984	0.9984	292.36	0.0347	Low / Med+
MLP	0.9989	0.9989	74.36	0.0107	Low / High
Voting	0.9989	0.9989	82.30	0.0203	Med- / Med

Table 3: Comparison of alternative stacking meta-classifiers: Accuracy (Acc.), F1-score (F1), Training and Inference time (in seconds), and interpretability vs. overfitting risk.

The overall architecture is designed to address the key challenges of modern intrusion detection systems, including scalability, adaptability to drifts, and high accuracy under real-time constraints. The integration of deep learning, ensemble methods, and meta-learning ensures that the proposed framework achieves a balance between robustness and efficiency, making it suitable for large-scale and continuously evolving data streams.

Model	Parameters / Configuration
Neural Network	Architecture: Dense(128, ReLU) → Dropout(50%) → Dense(64, ReLU) → Dropout(50%) → Dense(1, Sigmoid) Compilation: loss = binary_crossentropy, optimizer = adam (lr = 0.001), metric: accuracy Training: 100 epochs, batch size = 256
Random Forest	n_estimators = 200, max_depth = 15, criterion = gini, max_features = sqrt
LightGBM	min_gain_to_split = 0.1, min_child_samples = 20, min_data_in_leaf = 30, num_leaves = 40, max_depth = 15, learning_rate = 0.05, feature_fraction = 0.8, bagging_fraction = 0.8, bagging_freq = 5
Logistic Regression	max_iter = 20000, C = 1.0, penalty = l2, solver = lbfgs

Table 4: Hyperparameters and Configurations for the Hybrid Model Components.

3.4 Drift Handling

The drift handling implemented in this framework is based on an active adaptation module, which is designed to dynamically identify, monitor, and respond to data distribution shifts, a common challenge in such systems. The component initially works using a Hoeffding Drift Detection Method (HDDM) [22], which continuously monitors the incoming data for signs of drift, as depicted in the Mixed-Drift Handling Module in Figure 1. This statistical test can detect gradual and sudden changes in the data distribution by comparing the means of two consecutive sliding windows. A drift is triggered when the observed difference between the reference and current

windows exceeds a threshold ϵ , computed using the Hoeffding bound:

$$\epsilon = \sqrt{\frac{R^2 \cdot \ln(1/\delta)}{2n}}$$

where R is the range of the observed variable (typically set to 1 for normalized values), δ is the confidence parameter (we use $\delta = 10^{-7}$, equivalent to a 99.99% confidence level), and n is the number of samples in the sliding window. In our configuration, for a window size of $n = 1000$, the resulting threshold is approximately $\epsilon \approx 0.0898$, meaning a drift will trigger if a variation of 8.98% or greater is observed between the mean statistics of the reference and current windows. Thus, a drift is signalled when the absolute difference between the mean error rates of the two windows satisfies:

$$|\mu_1 - \mu_2| > \epsilon$$

Here, μ_1 and μ_2 are the means of the reference and current windows, respectively. This high-confidence setting ensures that only statistically significant changes are treated as drift events, thus reducing the likelihood of false positives in highly dynamic environments.

Upon detection, the system activates the Concept Drift Adaptation Module and/or the Feature Drift Detection Module. The former decides whether to apply fine-tuning or a full retraining of the model, depending on the severity of the drift. The latter recomputes Information Gain (IG) values and identifies which features are responsible for the shift, prompting selective retraining.

When full retraining is performed, an Adaptive Generative Model is triggered. This component is responsible for simulating new drifts using a Generative Adversarial Network (GAN) [23] to evaluate the robustness of the updated model and anticipate future retraining needs. Since GANs require a large and diverse set of training examples to generate realistic samples, we apply again SMOTE as a preprocessing step. This ensures a more balanced and representative set of attack samples before GAN training, mitigating the risk of generating unrealistic or biased outputs due to class imbalance. Algorithm 1 explains

how this component integrates into the framework using pseudocode.

Algorithm 1 Drift Handling Algorithm with an Active Adaptation Module

Require: Incoming data stream D
Ensure: Updated model M

```

1: Initialise model  $M$  and Mixed-Drift Handling Module
2: Set Hoeffding confidence parameter  $\delta = 10^{-7}$ , range  $R = 1$ 
3: while True do
4:   Observe new data batch  $B$  from  $D$ 
5:   Let  $n \leftarrow$  number of samples in  $B$ 
6:   Compute Hoeffding bound threshold ( $\epsilon$ )
7:   Estimate mean error in reference window:  $\mu_1$ 
8:   Estimate mean error in current window:  $\mu_2$ 
9:   if Drift Detected ( $|\mu_1 - \mu_2| > \epsilon$ ) then
10:    if Concept Drift Detected (model accuracy drops) then
11:      Activate Concept Drift Adaptation Module
12:      Compute severity of drift
13:      if Severe Drift then
14:        Perform full retraining of model  $M$ 
15:      else
16:        Apply fine-tuning to model  $M$ 
17:      end if
18:    end if
19:    if Feature Drift Detected (IG values changed) then
20:      Activate Feature Drift Detection Module
21:      Compute new Information Gain (IG) values for features
22:      Select features responsible for drift
23:      Retrain model  $M$  with the updated feature set
24:    end if
25:    if Full retraining executed then
26:      Trigger Adaptive Generative Model using GAN
27:      Simulate new drifts and test model resilience
28:      Perform additional retraining if required
29:    end if
30:  end if
31:  Update model  $M$  with  $B$ 
32: end while

```

4 Model evaluation

This section presents the results obtained from the experimental evaluation of the proposed framework. All experiments were carried out on a workstation equipped with an AMD Ryzen 7 5800H CPU (8 cores, 16 threads), 16 GB of DDR4 RAM, and an NVIDIA RTX 3060 GPU running Ubuntu 24.04 LTS. The software environment included Python 3.12.8 using key libraries such as scikit-learn (1.6.0), TensorFlow (2.18.0) with Keras (3.8.0) and NumPy (2.0.2), among others.

4.1 Model Performance

The first part of the experiment involved the designed multilayered hybrid model, which we tested across several scenarios depending on the selection of features, the pre-processing of the adjustment of the imbalances and the type of model used, as shown in Tables 5 and 6. All experiments were carried out using the CIC-IDS2017 dataset. The metrics evaluated are the accuracy of the model as the closeness between the predicted and actual classes, the macro F1 score as the arithmetic mean of all the per-class F1 scores, and the time as the time consumed by the OS to train each model with the provided batch of data.

A consistent observation across our tests is that incorporating SMOTE generally enhances performance. Similarly, the choice of feature selection method plays a critical role: while PCA tended to provide stable results across models, the Top 32 Information Gain approach yields mixed outcomes, illustrating a trade-off between dimensionality reduction and the preservation of discriminative features.

When examining individual models, we can clearly note clear trade-offs. Simpler models like k-NN model, offer the advantage of fast training times; however, they can lag in terms of classification robustness. Conversely, models such as RF and LGBM deliver high accuracy and F1 scores in many configurations; however, their performance is sensitive to the feature selection method employed. In particular, the standalone Neural Network demonstrated exceptional results in some settings (e.g., high accuracy with PCA) but suffered when faced with less optimal feature configurations.

Balancing these considerations, the final model selected for the proposed framework is the NN+LR[RF+LGBM] ensemble. This hybrid approach leverages the deep learning model’s ability to capture complex patterns and the ensemble’s strength in aggregating diverse predictive insights. Although NN+LR[RF+LGBM] can require longer training times in certain configurations, its consistently superior accuracy and macro F1 score makes it the most robust option for handling imbalanced data and variable feature spaces. In addition, the

		RF			k-NN			LGBM			NN		
		Acc	F1	Time (s)	Acc	F1	Time (s)	Acc	F1	Time (s)	Acc	F1	Time (s)
NO-SMOTE	PCA (37)	.9735	.5439	41.70	.9979	.8157	1.58	.7933	.0983	5.72	.9993	.9744	25.15
	Top 32 IG	.9912	.6312	3.16	.9642	.6161	1.59	.9341	.4080	5.81	.7995	.1269	26.02
SMOTE	PCA (37)	.9304	.9325	37.11	.9930	.9930	1.62	.9982	.9982	7.45	.9987	.9987	23.55
	Top 32 IG	.9845	.9840	6.04	.9632	.9627	1.60	.9980	.9980	9.58	.1433	.0358	24.58

Table 5: Performance of **RF**, **k-NN**, **LGBM**, and **NN** under different feature selection, dimensionality reduction, and oversampling settings.

		XGBoost			SVM			LR[RF+LGBM]			NN+LR[RF+LGBM]		
		Acc	F1	Time (s)	Acc	F1	Time (s)	Acc	F1	Time (s)	Acc	F1	Time (s)
NO-SMOTE	PCA (37)	.9987	.8850	1.03	.9988	.8314	3.22	.8512	.8512	267.15	.9420	.9380	312.95
	Top 32 IG	.9321	.8619	0.75	.8727	.3333	21.38	.9321	.9211	57.43	.7699	.7699	45.12
SMOTE	PCA (37)	.9972	.9972	1.12	.9985	.9985	3.83	.9985	.9985	235.21	.9989	.9989	285.40
	Top 32 IG	.9921	.9921	0.89	.5970	.5961	73.24	.9996	.9996	61.80	.9989	.9989	62.50

Table 6: Performance of **XGBoost**, **SVM**, **LR[RF+LGBM]**, and **NN+LR[RF+LGBM]** under different feature selection, dimensionality reduction, and oversampling settings.

selection of features based on Information Gain (IG) was the most optimal in terms of accuracy and computational time.

For our first iteration of the proposed framework, we chose IG over PCA for three main reasons:

1. Empirical Trade-Offs

- For our NN+LR[RF+LGBM] ensemble, SMOTE+IG and SMOTE+PCA both reach 99.99% accuracy and 0.9999 macro F1, but IG cuts training time from 285.4 s to 62.5 s.
- Other classifiers show that PCA’s tiny (< 0.2%) accuracy gains come at a 3–5× run-time penalty (e.g., RF with PCA: 0.9304/0.9325 in 37.11 s vs. IG: 0.9845/0.9840 in 6.04 s).

2. Computational Efficiency & Interpretability

- HDDAF must retrain quickly under drift and IG’s selection avoids the overhead of computing PCA components and extra matrix multiplications.

- IG preserves original feature semantics, which tree models exploit directly and makes it easier to interpret attack indicators, while PCA’s transformed axes obscure these relationships.

3. Synergy with Our Hybrid Architecture

- RF and LGBM naturally leverage IG’s raw-feature ranking for cleaner splits, whereas PCA mixes attributes and can weaken split quality.
- For the NN branch, IG-selected features maintain meaningful numerical patterns, while PCA’s rotated axes sometimes hinder convergence.

4.2 Framework Adaptability Evaluation

The second part of the experiment aimed to evaluate the performance of the HDDAF as a whole under various drift scenarios and adversarial attack conditions. To assess the overall performance of the

HDDAF framework under dynamic and challenging conditions, we designed a series of experiments to simulate various drift scenarios and adversarial attack conditions. In these experiments, the hybrid model was deployed within the full framework, while other modules remain unchanged from the previous section. The evaluation focused on the following three key objectives:

1. **Drift Detection and Recovery:** Measuring how quickly and effectively the system identifies and adapts to changes in the data, measured by the number of delayed batches until detection.
2. **Retraining Efficiency:** Comparing the computational cost and adaptation effectiveness of fine-tuning versus full retraining methods, measured using the retraining time and accuracy.
3. **Adversarial Robustness:** Evaluating the framework resilience when confronted with adversarial samples designed to poison the data, measured with the number of adversarial samples correctly detected and removed.

To simulate these conditions, data is streamed into the system in batches ranging from 2,000 to 5,000 samples. Each batch initially follows a baseline distribution, after which specific drift or adversarial scenarios are introduced at predetermined intervals.

Concept Drift scenarios mimic changes in the underlying relationship between features and labels:

- **Gradual Drift:** The decision boundary shifts slowly over several batches, indicating a gradual evolution of the concept. This was implemented by applying a 5-10% incremental change in feature-label correlation across 10-20 batches, gradually modifying the target distribution while maintaining continuity.
- **Sudden Drift:** A rapid, abrupt change is introduced at a specific batch to simulate an unexpected alterations in the data generation process. At a fixed batch index, samples are replaced with a new distribution with different statistical properties, which immediately shifts the decision boundaries.

- **Recurring Drift:** The data distribution alternates between different regimes in a cyclical pattern, reflecting environments where previously encountered concepts reoccur. This is achieved by cycling through previously observed distributions every 20-30 batches, simulating seasonal variations or periodic changes in user behaviour.
- **Incremental Drift:** Small, consistent modifications are applied over successive batches, leading to a steady, cumulative shift in the concept. This is introduced by applying slight, continuous transformations to the numerical features across all incoming batches, a small additive Gaussian noise in our case.

Feature Drift scenarios focus on changes at the feature level:

- **Feature Removal:** Specific features are removed from the data stream at set intervals, simulating scenarios such as failures or missing data. This process is executed by randomly masking or entirely dropping certain features from input data every 5-10 batches.
- **Feature Addition:** New features are introduced into the stream to simulate the integration of additional data sources. This process is performed by appending new synthetic or engineered features based on the transformations of existing features every 10-20 batches.
- **Correlation Change:** The inter-feature relationships were altered, affecting the overall structure and correlation patterns in the dataset. This is achieved by dynamically modifying the correlation coefficients between key features or reshuffling feature dependencies across different batches.

To test adversarial robustness, we simulate attacks by manually perturbing a subset of the samples. Each adversarial sample is crafted in three stages, to mimic realistic data-poisoning attacks:

1. *Feature-Space Perturbation.* Each incoming batch consists of 2,000–5,000 samples, creating

a set denoted as X with each sample denoted as x^i . For every sample in the batch, its features are represented by x_j , where $j \in F$, being F the complete set of feature indices. Thus, each x_j represents the value of the feature j for a given sample.

We randomly select 10% of X to poison. For each chosen sample, we pick 20% of its active (IG-selected) features from F and add a small noise term:

$$x_j^{i'} = x_j^i + \delta_j, \quad j \in S,$$

where S is a random subset of feature indices, and each δ_j is drawn uniformly from $[-\alpha\sigma_j, \alpha\sigma_j]$. Here, σ_j is the standard deviation of feature j over the training set, and α must be greater than 0, defining the amplitude (or radius) of the perturbation; in this example, we have selected $\alpha = 0.1$. By limiting perturbations to at most 10 % of each feature’s typical range, poisoned flows remain plausible, they do not become obvious outliers and are shifted enough to confuse the classifier.

2. *Label Flipping.* After perturbation, every modified flow is assigned an incorrect class label:

- If the original flow was “benign,” it is re-labelled as a randomly chosen attack family (one of the eight CIC-IDS2017 attack classes).
- If the original flow was an attack, it is re-labelled as “benign.”

This combination of mild feature-space noise plus a flipped label imitates a data-poisoning evasion attack, where an adversary crafts near-benign traffic to evade detection.

3. *Frequency and Timing.* We inject adversarial samples at fixed intervals, once every 5–20 batches, so that approximately 5%–1% of all streamed flows are poisoned overall. By randomizing both which flows and which features are perturbed in each batch, there is no optimization or gradient-based process guiding the noise.

Drift Type	Scenario	Delay (B)	Acc. Before (%)	Acc. After (%)	Retraining	Time (s)
Concept	Gradual	3	99.95	96.50	Fine-tuning	12.20
Concept	Sudden	2	99.95	97.75	Full Retrain	75.62
Concept	Recurring	3	99.80	98.22	Fine-tuning	13.50
Concept	Increm.	4	99.95	96.80	Fine-tuning	12.86
Feature	Removal	1	99.96	94.02	Full Retrain	74.10
Feature	Addition	2	99.78	96.15	Fine-tuning	18.05
Feature	Corr. Chg.	3	99.89	97.68	Full Retrain	76.78

Table 7: Drift Robustness Experimental Results.

Injecting at these intervals reflects a realistic attack cadence. Random selection ensures no fixed pattern, making it harder for the detector to memorize a static attack signature. By limiting the overall poisoning rate, we test HDDAF’s ability to identify and remove a small fraction of malicious flows without excessively retraining on clean data.

More advanced, optimization-based attacks like GAN-based evasion [7] are left for future work. This simple, reproducible technique is sufficient to demonstrate that HDDAF’s Hybrid Adversarial Training (clustering + RONI) can detect and remove up to 90% of these poisoned samples, as shown in Table 7 and Figure 8.

Each scenario was designed to reflect realistic conditions that a deployed system may face. By streaming the data in controlled batches and introducing these scenarios at predefined intervals, we can accurately monitor performance changes. The framework response is evaluated in terms of detection delay, accuracy before and after drift events, retraining time, and ability to withstand adversarial manipulation.

Table 7 provides an overview of the experimental results, summarising key metrics such as detection delays, pre- and post-drift accuracies, retraining methods, and times, as well as the measured adversarial robustness. Figure 7 represents box plots of model accuracies before and after drift for seven distinct scenarios, illustrating the range of performance for each scenario.

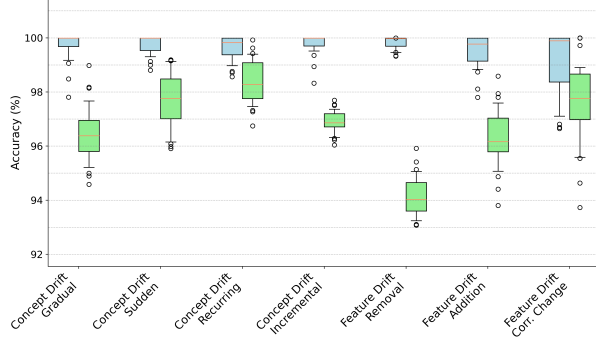


Figure 7: Boxplot Representation for Tests Performed in Table 7.

Figure 8 illustrates the effect of increasing poisoning percentages on model accuracy and Mean Squared Error (MSE), highlighting both the vulnerability of the baseline classifier and the effectiveness of the hybrid defence model explained before, consisting of a Hybrid Adversarial Training Technique.

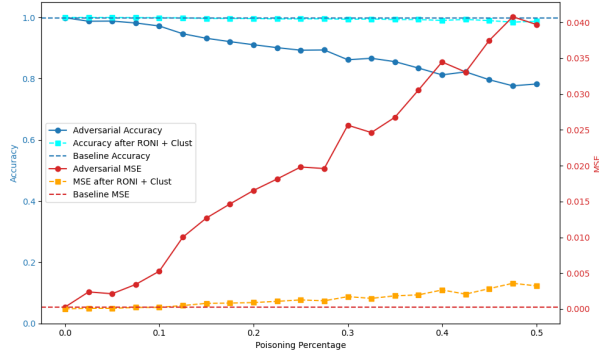


Figure 8: Evolution of Performance Metrics for a Poisoning Attack Scenario.

As the poisoning percentage increases, the adversarial accuracy declines steadily, whereas the adversarial MSE rises sharply, demonstrating the growing distortion introduced by poisoned data. However, after applying RONI and clustering, the model maintained near-baseline accuracy and a significantly

lower MSE, indicating successful mitigation of adversarial interference.

5 Discussion

In this section, we present a series of advanced experiments designed to evaluate the effectiveness and maintainability of the proposed HDDAF model as a whole using different datasets in three key aspects: generalization, module contribution, and real-time detection. For each aspect, we describe the experimental setup, report the obtained results, and discuss the implications of these findings.

5.1 Generalization

The goal of the generalization experiments is to assess how well the HDDAF model, trained on CIC-IDS2017, performs when applied to different yet related datasets. We evaluate the model on the CIC-IDS2018 and UNSW-NB15 datasets, measuring accuracy and training time to quantify any performance degradation. We use the full framework deployment for this task, and we will progressively ingest new data as if it were a drift event.

In Figure 9 (top panel), each ‘fold’ represents an abrupt drift in the data mixture (e.g. from 100% CIC-IDS2017 directly to an 80 / 20 blend of CIC-IDS2017 / CIC-IDS2018). For each fold, the model remains with the original configuration. The red circles trace the initial accuracy just before retraining, and the blue crosses show accuracy immediately afterward; the gray bars on the secondary axis report the time required for each retraining step. We see that, after a large drift (e.g. 60 / 40 or 40 / 60), initial accuracy can drop below 50%, but a single retraining restores it to over 85% at the cost of 60–65s of computation.

In the bottom panel of the same figure, we simulate abrupt drift again, but in this case we carry forward the retrained model into the next fold, so each retraining builds on the previous one. Here, accuracy degrades more gently (never below 0.83), and retraining times occur only when a new dataset is first introduced (at the 80 / 20 splits), reducing both performance loss and compute overhead.

Together, these experiments demonstrate that HDDAF can successfully generalize across evolving data distributions: abrupt retraining fully recovers performance at the expense of higher latency, while rolling adaptation balances accuracy and efficiency by limiting retraining to the most significant shifts.

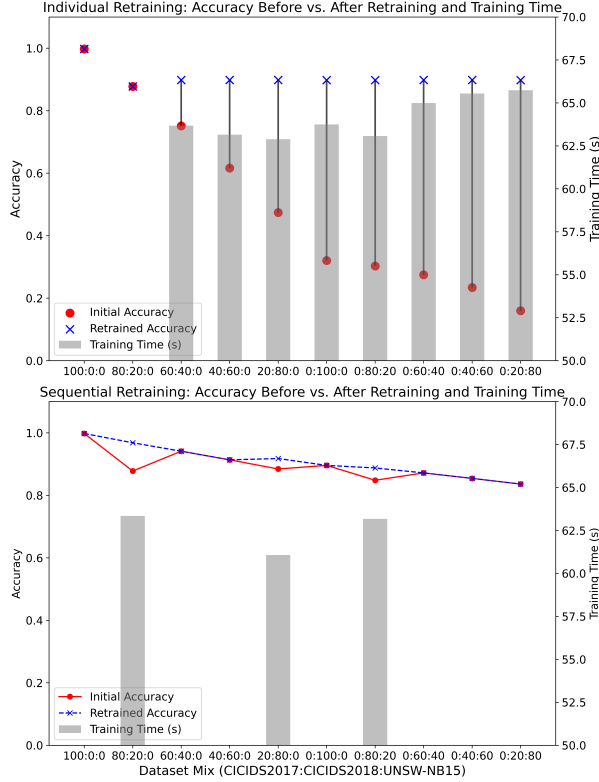


Figure 9: Comparison of HDDAF generalization under abrupt versus sequential retraining: **Top**: Individual retraining after each sudden change in dataset mix. **Bottom**: Sequential retraining building on the previously adapted model.

5.2 Contribution of the Modules

To quantify the individual contribution of each component in HDDAF, we perform an ablation study that systematically removes one or more modules and observes the resulting degradation in different perfor-

mance metrics. This approach allows us to attribute gains in robustness, drift responsiveness, and computational efficiency directly to the modules studied. By comparing each reduced configuration against the full framework, we can validate that every block is necessary to achieve HDDAF’s overall resilience and adaptability.

We process a controlled stream of 50 consecutive batches, each containing 2,000–5,000 flows sampled from the dataset CIC-IDS2017, and then introduce a gradual concept drift by switching to an 80:20 mix of CIC-IDS2017:CIC-IDS2018. Within each batch stream, 10% of flows are poisoned with adversarial perturbations, as described in Section 4.2. We then evaluate ten different framework variants, each defined by the absence of one or more of these four modules (as shown in Figure 1): Adversarial Training (Adv), Mixed-Drift Handling Module (Md), Concept/Feature Drift Adaptation Modules (Adp), and Adaptive Generative Model (Gen). For every variant we record:

1. Accuracy pre-drift (on the raw CIC-IDS2017 batch)
2. Accuracy post-drift (on the first 80 : 20 mixed batches)
3. Macro F_1 post-drift under adversarial attack
4. Drift-detection latency (time from drift onset to alarm)
5. Drift detection delay (number of batches until alarm)
6. Adversarial detection rate (proportion of adversarial samples detected)
7. Inference latency (ms per flow)
8. Total retraining time (s).

A total of ten configurations are summarized in Table 8. The table reports each configuration’s pre- and post-drift accuracy, post-drift F_1 score, drift detection latency and delay (in batches), adversarial detection rate, inference latency, and average retraining time. The results include standard deviations derived

from multiple experiment runs. A dash (–) indicates that a given metric is not applicable due to the corresponding module being disabled. As shown in Figure 1, when the Md. module is disabled, both the Adp. and Gen. modules become inaccessible. The results demonstrate how disabling individual components affects HDDAF’s ability to sustain high detection accuracy, adapt quickly to drift, and remain robust against adversarial inputs. The only notable exception is the removal of the Gen module, which results in a nearly negligible performance change under these tightly controlled, fixed-batch conditions. This outcome is expected, as the Gen module is primarily intended to forecast future drifts in more open-ended streaming scenarios, rather than the predefined batch-drift setting used in this evaluation.

5.3 Real-Time Detection

Finally, we evaluate HDDAF’s capability to operate under strict real-time constraints with a production-like stream of network flows in 100 batches (2,000–5,000 flows each). As shown in Table 9, the system ingested a total of 345,059 flows in 108.07 s, yielding an average latency of 1.0755 s per batch (≈ 0.312 ms per flow) and an end-to-end throughput of roughly 3,193 flows/s.

Metric	Value	Metric	Value
Total batches ingested	100	Total samples ingested	345 059
Total elapsed time (s)	108.07	Avg time per batch (ms)	1,075.54
Avg time per sample (ms)	0.3117	Estimated throughput (flows/s)	3 193

Table 9: Summary of the real-time ingestion experiment.

In Figure 10 we plot, over real elapsed time, both the size of each incoming batch and the instantaneous CPU utilization taking into consideration all CPUs available. We observe:

- **Throughput consistency:** Despite batch-to-batch size variability, the average processing time per batch remains stable (≈ 1 s), demonstrating that the NN+LR[RF+LGBM] ensemble can sustain several thousand flows per second without significant slowdown.
- **CPU load under control:** The CPU utilization using a single core hovers between 30% and 60% (depending on the host machine’s core count), leaving sufficient headroom for parallel tasks or additional pre/post-processing.
- **Bottleneck identification:** Peaks in CPU usage coincide with the largest batches), suggesting opportunities to balance load via smaller, more frequent batches or by offloading parts of the inference pipeline to specialized hardware.

The real-time experiments show that processing very large batches at once can cause sudden spikes in CPU usage and increased latency. If these large batches are split into smaller groups of about one to two thousand flows each, the computation load becomes more evenly distributed. This approach minimizes resource spikes and helps maintain a steady end-to-end latency.

Overlapping stages of the pipeline, such as feature extraction, noise injection, and model inference, through asynchronous I/O or multithreaded workers can conceal data-fetch delays behind active computation. Converting model components to lower-precision formats, for instance, using 16-bit or 8-bit representations for LGBM and neural network weights, can reduce per-sample inference time by about 10 to 20 percent without significantly harming detection accuracy. Applying these strategies will enable HDDAF to sustain high throughput and low latency under production conditions, demonstrating its suitability for operational cybersecurity environments.

5.4 Key Insights

The key insights derived from these results are as follows:

- PCA-based models provide better accuracy but need higher computational requirements because of the complexity of the reduction. In addition, explainability is compromised as the features are transformed.

Configuration	Accuracy			Drift Detection		Robustness		Train Time (s)
	Pre Drift	Post Drift	F ₁ Post Drift	Lat. (ms)	Delay (Batches)	Adv. Rate	Inference Lat. (ms)	Avg. Time
Full framework	0.94 ± 0.02	0.90 ± 0.01	0.91 ± 0.02	450 ± 50	3 ± 1	0.92 ± 0.03	150 ± 10	74.5
-Adv	0.85 ± 0.01	0.75 ± 0.01	0.74 ± 0.02	470 ± 50	2 ± 1	–	150 ± 10	75.3
-Md(-Adp-Gen)	0.93 ± 0.02	0.65 ± 0.01	0.63 ± 0.01	–	–	0.91 ± 0.02	145 ± 10	74.2
-Adv-Md(-Adp-Gen)	0.83 ± 0.01	0.44 ± 0.01	0.23 ± 0.01	–	–	–	135 ± 10	–
-Adp	0.94 ± 0.01	0.68 ± 0.01	0.64 ± 0.01	400 ± 50	3 ± 1	0.92 ± 0.02	150 ± 8	74.2
-Gen	0.93 ± 0.01	0.90 ± 0.01	0.90 ± 0.01	300 ± 50	3 ± 1	0.90 ± 0.02	152 ± 10	76.5
-Adp-Gen	0.92 ± 0.02	0.67 ± 0.01	0.65 ± 0.01	420 ± 40	2 ± 1	0.91 ± 0.02	147 ± 10	–
-Adv-Adp	0.84 ± 0.01	0.51 ± 0.02	0.50 ± 0.02	400 ± 50	3 ± 1	–	150 ± 10	70.8
-Adv-Gen	0.86 ± 0.03	0.77 ± 0.01	0.75 ± 0.01	380 ± 50	4 ± 1	–	145 ± 8	73.5
-Adv-Adp-Gen	0.86 ± 0.02	0.45 ± 0.02	0.44 ± 0.01	400 ± 50	2 ± 1	–	138 ± 10	–

Table 8: Ablation study results measuring different metrics per configuration.

- Methods using IG information were more efficient but underperformed in comparison to PCA approaches.
- Combining models (e.g., through LR[RF+LGBM] and NN+LR[RF+LGBM]) provides a balanced solution that enhances both accuracy and robustness. Although these ensembles require greater computational resources, they consistently outperform individual models in handling class imbalances and feature drift.
- The Mixed-Drift Handling Module is effective in balancing model stability and efficiency. By using Hoeffding Drift Detection, the system dynamically decides between fine-tuning and full retraining based on drift severity, ensuring minimal disruption to ongoing operations. The accuracies obtained after drift consistently remain similar to previous ones.
- Hybrid adversarial training combines clustering-based outlier detection and RONI sanitisation, achieving a good balance in terms of accuracy and MSE when tested on adversarial poisoning attack scenarios.
- The modular design of the framework ensures scalability and adaptability. The ability of the proposed method to integrate incremental updates, semi-supervised learning, and cross-domain applications makes it a future-proof solution for real-world anomaly detection.
- Generalization experiments demonstrate that HDDAF can adapt to abrupt drifts, including different dataset mixes. Retraining process fully restores accuracy at the cost of higher latency, while rolling adaptation maintains smoother performance with fewer retraining steps.
- The ablation study confirms the necessity of each module in order to achieve overall robustness and fast drift recovery. Only the Generative Model has negligible effect under fixed-batch tests, reflecting its role in forecasting future drifts rather than immediate batch adaptation.
- Real-time detection tests show that HDDAF sustains over 3,000 flows/s with stable batch latencies and moderate CPU load. Splitting large batches into smaller chunks and asynchronous pipeline execution can further reduce spikes and tail latencies.
- Comparing HDDAF to other methods, we observe that it offers distinct advantages. It surpasses current available approaches such as HCDD-SHAP [31] and DFA-GPE [21] in both concept and feature drift handling. The proposed design ensures scalability and adaptability, making it suitable for real-time deployment

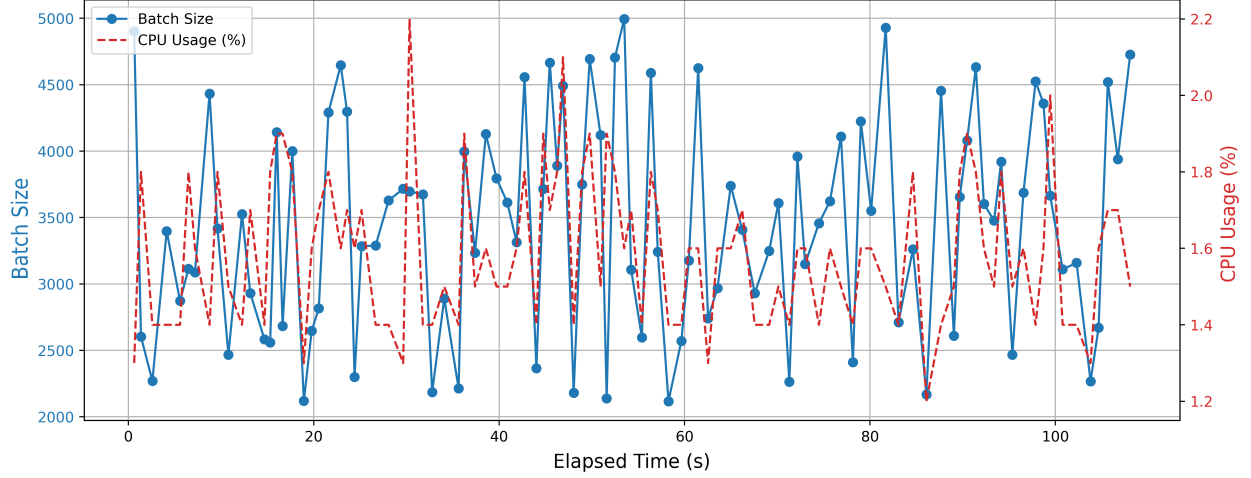


Figure 10: Real-time ingestion performance metrics.

in resource-limited environments.

5.5 Limitations

While the proposed HDDAF demonstrates competitive performance across multiple attack categories, several limitations should be considered:

- The framework is trained on static snapshots of past attacks, which limits its ability to generalize to zero-day or adaptive threats.
- Its multi-layered architecture improves detection but introduces computational overhead that may hinder real-time or resource-constrained deployment.
- Feature selection relies on filter-based methods (e.g., Information Gain) for efficiency, but wrapper or embedded approaches could offer better performance. Their exclusion is acknowledged and proposed as future work.
- The use of multiple classifiers and feature selectors reduces interpretability and increases sensitivity to hyperparameter tuning, affecting maintainability.

- Adversarial robustness is managed via sample rejection rather than a formal robust loss function. Incorporating such formulations remains an important direction for improvement.
- The CIC-IDS2017 and CIC-IDS2018 datasets used are known to contain annotation errors and artefacts [15, 16], which could impact evaluation reliability.
- The framework does not yet distinguish between benign and malicious drift. Legitimate service changes may be flagged as threats, suggesting the need for drift explainability and semi-supervised labelling strategies.

In light of these limitations, the future work should aim to enhance the adaptability of the framework, scalability, and transparency, while validating its effectiveness in dynamic and heterogeneous environments.

6 Conclusion

This paper presents the Hybrid Drift Detection and Adaptation Framework (HDDAF), a modular and

adaptive system that addresses concept drift, feature drift, and adversarial resilience in AI-based threat detection. By combining deep learning and ensemble models, HDDAF achieves higher robustness and adaptability than existing methods. The Mixed-Drift Handling Module enables the system to choose between fine-tuning and full retraining depending on the severity of drift, balancing accuracy and retraining cost.

Experimental results using CIC-IDS2017 showed that HDDAF consistently maintains macro F1 scores above 99.9 percent, outperforming established baselines such as HCDD-SHAP and DFA-GPE. Additional evaluations on CIC-IDS2018 and UNSW-NB15 confirmed HDDAF’s generalization capabilities, while ablation studies highlighted the importance of each module, especially adversarial training and active drift adaptation. The ensemble NN+LR[RF+LGBM] model proved most effective across multiple scenarios. SMOTE further improved classification performance under class imbalance, and IG-based feature selection delivered a better balance between interpretability, speed, and accuracy than PCA. Real-time testing confirmed that HDDAF processes over 3,000 flows per second with controlled latency and efficient resource usage.

Future work will focus on several key directions. First, incorporating semi-supervised learning could improve drift adaptation in environments with limited labelled data. Second, efforts to reduce computational cost, through lighter model variants and hardware-aware optimizations, are necessary for real-time and edge deployments. Third, integrating explainable AI techniques would enhance transparency and help analysts interpret decisions in critical settings or audit model updates. Fourth, exploring more adaptive or model-integrated feature selection strategies, including wrapper-based or embedded approaches, could further improve detection performance under dynamic drift conditions. Lastly, applying HDDAF to other dynamic domains such as healthcare anomaly detection and industrial monitoring will help assess its adaptability beyond cybersecurity.

Overall, HDDAF represents an advance in anomaly detection, proving that a hybrid and modular ap-

proach can deliver both high accuracy and practical adaptability. While deployment challenges remain, particularly in terms of efficiency and adversarial defence, this work lays the foundation for resilient, adaptive systems suited to real-world operational environments.

Conflict of Interests

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

Acknowledgements

This publication is part of the project “CiberIA: Investigación e Innovación para la Integración de Ciberseguridad e Inteligencia Artificial” (Proyecto C079/23), financed by “European Union NextGeneration-EU, the Recovery Plan, Transformation and Resilience”, through INCIBE. It has also been partially supported by the project SecAI (PID2022-139268OB-I00) funded by the Spanish Ministerio de Ciencia e Innovación, and Agencia Estatal de Investigación. Funding for open access charge: Universidad de Málaga / CBUA.

References

- [1] M. Abdelaty, S. Scott-Hayward, R. Doriguzzi-Corin, and D. Siracusa. Gadot: Gan-based adversarial training for robust ddos attack detection. 2022. doi: 10.48550/ARXIV.2201.13102. URL <https://arxiv.org/abs/2201.13102>.
- [2] H. Abdi and L. J. Williams. Principal component analysis. *WIREs Computational Statistics*, 2(4): 433–459, July 2010. ISSN 1939-0068. doi: 10.1002/wics.101. URL <http://dx.doi.org/10.1002/wics.101>.
- [3] G. J. Aguiar and A. Cano. Enhancing concept drift detection in drifting and imbalanced data streams through meta-learning. In *2023 IEEE International Conference on Big Data (BigData)*. IEEE, Dec. 2023. doi: 10.1109/bigdata59044.2023.10386364. URL <http://dx.doi.org/10.1109/BigData59044.2023.10386364>.

- [4] J. P. Anderson. Computer security threat monitoring and surveillance. Technical report, James P. Anderson Co., Fort Washington, Pennsylvania, 1980. Technical report prepared for the U.S. Air Force Electronic Systems Division.
- [5] J. P. Barddal, H. M. Gomes, F. Enembreck, and B. Pfahringer. A survey on feature drift adaptation: Definition, benchmark, challenges and future directions. *Journal of Systems and Software*, 127:278–294, 2017. ISSN 0164-1212. doi: <https://doi.org/10.1016/j.jss.2016.07.005>. URL <https://www.sciencedirect.com/science/article/pii/S0164121216301030>.
- [6] M. Buda, A. Maki, and M. A. Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106:249–259, 2018. doi: [10.1016/j.neunet.2018.07.011](https://doi.org/10.1016/j.neunet.2018.07.011). URL <https://www.sciencedirect.com/science/article/pii/S0893608018302107>.
- [7] Z. Cai, Z. Xiong, H. Xu, P. Wang, W. Li, and Y. Pan. Generative adversarial networks: A survey toward private and secure applications. *ACM Computing Surveys*, 54(6):1–38, 2021. doi: [10.1145/3459992](https://doi.org/10.1145/3459992).
- [8] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, June 2002. ISSN 1076-9757. doi: [10.1613/jair.953](https://doi.org/10.1613/jair.953). URL <http://dx.doi.org/10.1613/jair.953>.
- [9] D. E. Denning. An intrusion-detection model. *IEEE Transactions on Software Engineering*, SE-13(2): 222–232, 1987. doi: [10.1109/TSE.1987.232894](https://doi.org/10.1109/TSE.1987.232894).
- [10] K. A. ElDahshan, A. A. AlHabshy, and B. I. Hameed. Meta-heuristic optimization algorithm-based hierarchical intrusion detection system. *Computers*, 11(12):170, Nov. 2022. ISSN 2073-431X. doi: [10.3390/computers11120170](https://doi.org/10.3390/computers11120170). URL <http://dx.doi.org/10.3390/computers11120170>.
- [11] R. Fan, M. Zhong, S. Wang, Y. Zhang, A. Andrew, M. Karagas, H. Chen, C. Amos, M. Xiong, and J. Moore. Entropy-based information gain approaches to detect and to characterize gene-gene and gene-environment interactions/correlations of complex diseases. *Genetic Epidemiology*, 35(7):706–721, Oct. 2011. ISSN 0741-0395. doi: [10.1002/gepi.20621](https://doi.org/10.1002/gepi.20621). URL <http://dx.doi.org/10.1002/gepi.20621>.
- [12] A. Halbouni, T. S. Gunawan, M. H. Habaebi, M. Halbouni, M. Kartiwi, and R. Ahmad. Cnn-lstm: Hybrid deep neural network for network intrusion detection system. *IEEE Access*, 10:99837–99849, 2022. ISSN 2169-3536. doi: [10.1109/access.2022.3206425](https://doi.org/10.1109/access.2022.3206425). URL <http://dx.doi.org/10.1109/ACCESS.2022.3206425>.
- [13] R. Harwahyu, F. H. Erasmus Ndolu, and M. V. Overbeek. Three layer hybrid learning to improve intrusion detection system performance. *International Journal of Electrical and Computer Engineering (IJECE)*, 14(2):1691, Apr. 2024. ISSN 2088-8708. doi: [10.11591/ijece.v14i2.pp1691-1699](https://doi.org/10.11591/ijece.v14i2.pp1691-1699). URL <http://dx.doi.org/10.11591/ijece.v14i2.pp1691-1699>.
- [14] H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009.
- [15] M. Lanvin, P.-F. Gimenez, Y. Han, F. Majorczyk, L. Mé, and É. Totel. Errors in the cids2017 dataset and the significant differences in detection performances it makes. In *International Conference on Risks and Security of Internet and Systems (CRISIS)*, Lecture Notes in Computer Science, Cham, 2022. Springer Nature Switzerland. Partially supported by the Cyber Excellence Pole (PEC: DGA, Brittany Region).
- [16] L. Liu, G. Engelen, T. Lynar, D. Essam, and W. Joosen. Error prevalence in nids datasets: A case study on cic-ids-2017 and cse-cic-ids-2018. In *Proceedings of the 2022 IEEE Conference on Communications and Network Security (CNS)*, pages 254–262, Austin, TX, USA, 2022. IEEE. doi: [10.1109/CNS56536.2022.9969755](https://doi.org/10.1109/CNS56536.2022.9969755).
- [17] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang. Learning under Concept Drift: A Review. *IEEE transactions on knowledge and data engineering*, page 1, 1 2018. doi: [10.1109/tkde.2018.2876857](https://doi.org/10.1109/tkde.2018.2876857). URL <https://doi.org/10.1109/tkde.2018.2876857>.
- [18] J. G. Moreno-Torres, T. Raeder, R. Alaiz-Rodríguez, N. V. Chawla, and F. Herrera. A unifying view on dataset shift in classification. *Pattern Recognition*, 45(1):521–530, Jan. 2012. ISSN 0031-3203. doi: [10.1016/j.patcog.2011.06.019](https://doi.org/10.1016/j.patcog.2011.06.019). URL <http://dx.doi.org/10.1016/j.patcog.2011.06.019>.

- [19] D. Mulimani, P. Patil, S. Totad, and R. Benni. On-line detection and adaptation of concept drift in streaming data classification. *Procedia Computer Science*, 235:2803–2811, 2024. ISSN 1877-0509. doi: 10.1016/j.procs.2024.04.265. URL <http://dx.doi.org/10.1016/j.procs.2024.04.265>.
- [20] B. Nelson, M. Barreno, F. J. Chi, A. D. Joseph, B. I. P. Rubinstein, U. Saini, C. Sutton, J. D. Tygar, and K. Xia. Exploiting machine learning to subvert your spam filter. In *Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, LEET’08, USA, 2008. USENIX Association.
- [21] M. S. Noori, R. K. Z. Sahbudin, A. Sali, and F. Hashim. Feature drift aware for intrusion detection system using developed variable length particle swarm optimization in data stream. *IEEE Access*, 11:128596–128617, 2023. ISSN 2169-3536. doi: 10.1109/access.2023.3333000. URL <http://dx.doi.org/10.1109/ACCESS.2023.3333000>.
- [22] A. Pesaranghader and H. L. Viktor. *Fast Hoeffding Drift Detection Method for Evolving Data Streams*, page 96–111. Springer International Publishing, 2016. ISBN 9783319462271. doi: 10.1007/978-3-319-46227-1_7. URL http://dx.doi.org/10.1007/978-3-319-46227-1_7.
- [23] M. Ring, D. Schlör, D. Landes, and A. Hotho. Flow-based network traffic generation using generative adversarial networks. *Computers and Security*, 82:156–172, May 2019. ISSN 0167-4048. doi: 10.1016/j.cose.2018.12.012. URL <http://dx.doi.org/10.1016/j.cose.2018.12.012>.
- [24] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*, pages 108–116, January 2018. doi: 10.5220/0006639801080116. URL <https://doi.org/10.5220/0006639801080116>.
- [25] S. Sheikhi and P. Kostakos. A novel anomaly-based intrusion detection model using psogwo-optimized bp neural network and ga-based feature selection. *Sensors*, 22(23):9318, Nov. 2022. ISSN 1424-8220. doi: 10.3390/s22239318. URL <http://dx.doi.org/10.3390/s22239318>.
- [26] A. V. Turukmane and R. Devendiran. M-multisvm: An efficient feature selection assisted network intrusion detection system using machine learning. *Computers and Security*, 137:103587, Feb. 2024. ISSN 0167-4048. doi: 10.1016/j.cose.2023.103587. URL <http://dx.doi.org/10.1016/j.cose.2023.103587>.
- [27] T. Wisanwanichthan and M. Thammawichai. A double-layered hybrid approach for network intrusion detection system using combined naive bayes and svm. *IEEE Access*, 9:138432–138450, 2021. ISSN 2169-3536. doi: 10.1109/access.2021.3118573. URL <http://dx.doi.org/10.1109/ACCESS.2021.3118573>.
- [28] D. Wishart. *k-Means Clustering with Outlier Detection, Mixed Variables and Missing Values*, page 216–226. Springer Berlin Heidelberg, 2003. ISBN 9783642557217. doi: 10.1007/978-3-642-55721-7_23. URL http://dx.doi.org/10.1007/978-3-642-55721-7_23.
- [29] Q. Xiang, L. Zi, X. Cong, and Y. Wang. Concept Drift Adaptation Methods under the Deep Learning Framework: A Literature Review. *Applied Sciences*, 13(11):6515, 5 2023. doi: 10.3390/app13116515. URL <https://doi.org/10.3390/app13116515>.
- [30] X. Yuan, S. Han, W. Huang, H. Ye, X. Kong, and F. Zhang. A simple framework to enhance the adversarial robustness of deep learning-based intrusion detection system. *Computers & Security*, 137:103644, Feb. 2024. ISSN 0167-4048. doi: 10.1016/j.cose.2023.103644. URL <http://dx.doi.org/10.1016/j.cose.2023.103644>.
- [31] H. Zhang, X. Chen, M. Hu, and V. Sugumaran. Multilayer concept drift detection method based on model explainability. *IEEE Access*, page 1–1, 2024. ISSN 2169-3536. doi: 10.1109/access.2024.3517697. URL <http://dx.doi.org/10.1109/ACCESS.2024.3517697>.
- [32] X. Zhao, K. W. Fok, and V. L. Thing. Enhancing network intrusion detection performance using generative adversarial networks. *Computers & Security*, 145:104005, Oct. 2024. ISSN 0167-4048. doi: 10.1016/j.cose.2024.104005. URL <http://dx.doi.org/10.1016/j.cose.2024.104005>.