Real-time Intrusion Detection in IoT: Employing Adaptive Models and Hardware Acceleration

PhD Dissertation Proposal

by

MsC. Eric García Huitzitl

Doctoral Advisors:

Ph.D. René Armando Cumplido Parra, INAOE

Ph.D. Lázaro Bustio Martínez, IBERO

Instituto Nacional de Astrofísica, Óptica y Electrónica

©Coordinación de Ciencias Computacionales

August, 2024

Santa María de Tonantzintla, Puebla, CP 72840

Contents

Abstract

The evolving landscape of IoT-generated data presents significant challenges, particularly with concept drift, where shifts in statistical distributions within data streams can become cybersecurity threats. Traditional machine learning, relying on static models, struggles to effectively address concept drift highlighting the necessity for adaptive models tailored to streaming data. This paper explores strategies for enhancing IoT security in dynamic data environments. A hybrid concept drift detection method that combines error rate analysis with data distribution monitoring is proposed. Sliding window-based data capture and drift analysis, coupled with K-means clustering and Random Forest, enhance the training dataset, incorporating both fixed and adaptive windows. Adaptive Random Forest is employed for anomaly detection and model retraining. Experimental trials were conducted using the NSL-KDD and IoTID20 datasets to evaluate the model performance. Principal Component Analysis (PCA) and Spearman's correlation coefficient are used to detect and quantify concept drift. The proposed adaptive model demonstrates significant improvements, with Adaptive Random Forest achieving 98.66% and 99.69% accuracy, 99.52% and 99.70% precision, 97.74% and 99.75% detection rate, 98.78% and 99.83% F1-score, and a low false alarm rate of 1.45% and 4.32%, respectively. However,

experimental results show that adaptation time exceeds prediction time, potentially impairing model performance in real-time intrusion detection systems. Thus, investigating hardware acceleration techniques and algorithmic optimization is crucial to improve system performance.

Adaptive IDS Hybrid Approach Concept Drift Clustering Classification

1 Introduction

In recent years, the number of devices connected to the internet has surged dramatically, accompanied by a rapid increase in data volume. According to [Statista, 2024], up to 39.6 billion IoT devices are projected to be connected to the internet by 2033. However, alongside this rapid development, the number of network attacks for various purposes has also increased in recent years. In 2023, Kaspersky reported detecting around 411,000 malicious files, a nearly 3% increase from the previous year [Kaspersky, 2023]. Most brute-force password attempts (97.91%) targeted Telnet, with only 2.09% focusing on SSH. The identified malicious files included DDoS Botnets, Ransomware, Miners, DNS Changers, and Proxy Bots, which compromise system security by overloading servers, encrypting data for ransom, mining cryptocurrencies, redirecting traffic, or concealing malicious activities. Therefore, to mitigate these evolving threats, advanced security measures are essential [L. Yang and Shami, 2021].

In addressing this need, `INTRUSION DETECTION SYSTEMS (IDS)` stand out as fundamental tools. An IDS is a security tool designed to monitor and analyze network traffic or system activities for suspicious or unauthorized behavior that may indicate a potential intrusion or attack [Yahyaoui et al., 2021]. IDS can be divided into two main approaches: a) Signature-based detection and b) Anomaly-based detection [Roshan et al., 2018]. Signature-based detection compares network traffic

against a database of known attacks signatures and malicious behaviors. While signature-based IDS are notably swift and efficient at identifying known threats, their reliance on pre-existing data renders them ineffective at detecting unknown attacks, such as zero-day attacks, wherein assailants exploit vulnerabilities that have not been previously identified [Roshan et al., 2018]. On the other hand, anomaly-based detection discerns unusual patterns or behaviors in network traffic or system activities that stray from a predefined baseline of normal behavior [L. Yang and Shami, 2021]. This approach involves establishing a model of typical activity, often utilizing machine learning techniques, and subsequently utilizing this model to predict new instances [Seth, Singh, and Chahal, 2021]. Anomaly-based detection shines in identifying never-before-seen threats [Xiaolan et al., 2022]. This makes it appealing for cybersecurity researchers as it can uncover new attack vectors that traditional signature-based methods might miss.

However, the `concept drift` phenomenon, which signifies shifts in the relationship between input and output variables over time within predictive models, is a crucial concern in the cybersecurity domain, particularly in IoT, where device heterogeneity and diverse physical environments significantly influence it [L. Yang and Shami, 2021]. The real-time deployment of IDS is especially vulnerable to false positives due to the dynamic nature of cyber threats, which frequently alter the patterns and behaviors these systems are designed to detect [Babüroğlu, Durmuşoğlu, and Dereli, 2023]. For instance, Figure 1 reveals fluctuations in time, energy, traffic level, and irregular patterns, characteristics typically associated with attacks. However, this distribution may not always remain constant. A firmware update can alter data distribution, resembling a DDoS attack, and cause misclassification if the model isn't updated. This increase in false alarms and decline in detection accuracy forces network administrators to seek alternative tools and solutions.

Developing adaptive models is crucial for adjusting to emerging patterns in dynamic data streams [Seth, Singh, and Chahal, 2021]. Incremental learning algo-

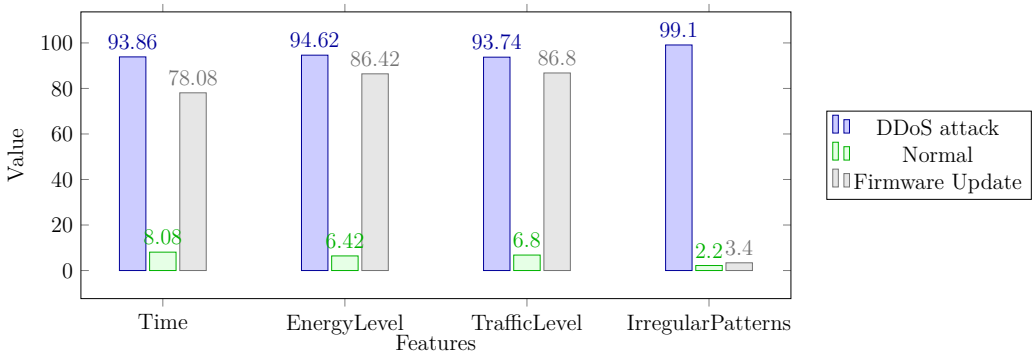


Figure 1: Data Distribution for Normal, DDoS Attacks, and Firmware Updates.

rithms, which continuously update models with new data, and concept drift detection methods are essential for maintaining model accuracy and relevance in dynamic environments [L. Yang and Shami, 2021]; [Jain and Kaur, 2021]. Additionally, employing methods that do not solely rely on labeled datasets is necessary for effective detection in environments with limited labeled data, thereby maintaining IDS sensitivity to evolving cyber threats and enabling real-time attack detection [Jain, Kaur, and Saxena, 2022]. Finally, given the limited computational resources and memory constraints of IoT devices, which restrict their ability to handle large volumes of data and complex learning models, it is crucial to develop adaptive models with low complexity [L. Yang and Shami, 2021].

This proposal presents an adaptive model for IDS that includes a method for detecting concept drift. The model aims to improve the performance and reliability of IDS by dynamically adjusting to changes in data patterns over time. A hybrid approach, combining K-Means and Random Forest, is employed to manage concept drift. Additionally, Adaptive Random Forest is utilized for model updating and attack detection. A hybrid concept drift detection method, based on distributed data and error rates and employing a window-based technique, is proposed for identifying concept drift. This method leverages Principal Component Analysis (PCA) and the Spearman correlation coefficient to detect and quantify the severity of concept drift. The effectiveness of the proposed adaptive IDS is rigorously evaluated using vari-

ous datasets, with comparisons made to recent research. Established performance metrics are systematically applied to assess the efficacy of the proposed IDS.

The contributions of this proposal document can be summarized as follows:

- ◇ Introduced a hybrid concept drift detection method that combines distribution analysis and statistical metrics with an independent error rate-based method, where the error rate-based method enhances the overall handling of concept drift.
- ◇ Developed a hybrid unsupervised approach with incremental learning for IDS to manage concept drifts without needing a labeled dataset, allowing model updates only when necessary and thereby reducing resource consumption.

The preliminary results demonstrate the effectiveness of the proposed adaptive model and concept drift detection method. The Adaptive Random Forest achieved classification accuracies of 98.66% on the NSL-KDD dataset and 99.83% on the IoTID20 dataset. It also attained precision rates of 99.52% and 99.75%, detection rates of 97.74% and 99.75%, and F1-scores of 99.78% and 99.83%, respectively, while maintaining low false alarm rates of 1.14% and 4.32%. However, adaptation time is significantly longer than prediction time, making real-time IDS implementation impractical since inference cannot be paused. This results in suboptimal attack detection. The findings highlight the need for seamless transitions between model adaptation and responsiveness, and underscore potential performance degradation without timely adaptation and hardware acceleration.

Subsequent sections include a literature review in Section 2, the problem statement and objectives in Section 3, and the methodology in Section 4. Section 5 presents the adaptive model and concept drift detection method for IDS, along with preliminary results. The work plan is detailed in Section 6, and references are provided at the end of the document.

2 Background

IDS are essential for network administrators, providing effective traffic analysis and monitoring. They issue alerts for potential attacks, thereby protecting network integrity [Abdualrahman and Ibrahem, 2021]. However, the growing number of IoT devices and their diverse data present challenges to traditional IDS, particularly those that rely on machine learning and static models with large static datasets [García and López, 2024]. These systems face difficulties with `CONCEPT DRIFT`, especially in IoT environments where dynamic data patterns, such as firmware updates and changing conditions, complicate threat detection [Gómez and Rodríguez, 2024].

Adaptive models using incremental learning address concept drift by continually updating their knowledge, allowing IDS to adapt to the rapidly evolving IoT landscape. Integrating incremental learning and concept drift detection enhances real-time threat detection and response, thereby providing robust defense against security breaches in the dynamic IoT environment [Seth, Singh, and Chahal, 2021]. The following sections will discuss related work on concept drift detection and adaptive models, summarizing existing approaches, and recent advancements in the field.

2.1 Methods for Concept Drift Detection in Continuous Data Streams

`Concept drift` refers to the shift in the statistical properties of input features and/or the target variable over time, can significantly impact predictive model performance by altering the relationship between these features and the target variable, which may lead to a decline in model accuracy [Lu et al., 2018]; [L. Yang and Shami, 2021].

[Jain, Kaur, and Saxena, 2022] divides the detection of concept drift in data streams into two approaches: (1) data distribution-based and (2) error rate-based.

Data distribution-based methods monitor changes in data distribution over

time by analyzing statistical properties, which help in detecting concept drift since the measurements used indicate differences between data samples [Seth, Singh, and Chahal, 2021]. For this purpose, they rely on `window-based methods`, particularly sliding windows, which serve as snapshots in time, denoted as t_0. These snapshots facilitate the comparison of distributions P_{t_0} and P_{t_1}, enabling the detection of changes in data patterns (concept drift) [L. Yang and Shami, 2021].

[Qahtan et al., 2015] proposed using PCA to detect abrupt changes in unlabeled multidimensional data streams. They used a Kullback-Leibler change score to measure variation between consecutive windows, aggregated from PCA-selected components. This method significantly reduces computational costs but may struggle with subtle or gradual changes, potentially leading to delayed or missed detection.

[Wahab, 2022] proposed using PCA to compare feature variance between windows. Drift is detected if the angle between eigenvalues is $60°$, indicating significant divergence. The method's drawback is its sensitivity to the threshold angle, which can cause false positives or missed detections if not well-calibrated.

[Chu et al., 2024] introduced an ensemble concept drift localization algorithm using nonparametric statistical methods (Kolmogorov-Smirnov, Wilcoxon rank-sum, and Mann-Kendall tests) to compare cumulative distribution functions of two samples. Drift is detected by significant differences (p-values below a threshold) between two windows. While effective across data distributions, this method is computationally intensive, increasing processing time and resource use.

However, despite their utility, fixed sliding windows have significant drawbacks: they may not effectively capture gradual changes in data distribution and can be inefficient due to their static nature [Seth, Singh, and Chahal, 2021]; [L. Yang and Shami, 2021]. A large window might incorporate outdated information, while a smaller window may lack sufficient context to accurately reflect changes. Conversely, adaptive windowing methods, such as Adaptive Windowing (ADWIN), dynamically

adjust the window size based on changes in data distribution. ADWIN, for example, uses variance as the primary criterion for adjustment, allowing for a more flexible response to evolving data patterns [Seth, Singh, and Chahal, 2021].

Error rate-based methods detect concept drift by utilizing different performance metrics to monitor changes in model performance [L. Yang and Shami, 2021]. By continuously tracking these metrics, these methods can signal when a model's predictions deviate from expected outcomes, thus indicating potential changes in the underlying data distribution.

[Jain and Kaur, 2021] proposed an error rate-based concept drift detection method monitoring accuracy and false alarm rate. Drift is indicated if accuracy drops below a threshold and the false alarm rate exceeds another threshold, signaling the need for model updates. Although this method does not measure drift severity directly, accuracy decreases can serve as an indirect indicator.

[L. Yang and Shami, 2021] proposed the Optimized Adaptive Sliding Windowing (OASW) method, combining sliding and adaptive window techniques with error-based methods. It retains data samples in the sliding window when accuracy drops, creating an adaptive window for model retraining. This method enables both drift detection and adaptation through its dual-window approach.

[Mahdi et al., 2023] proposed a drift detection method combining a disagreement measure with the Page-Hinkley test. The disagreement measure assesses the ratio of accurate versus inaccurate classifier observations, while the Page-Hinkley test tracks cumulative changes to detect drift. This method is computationally efficient but requires fully supervised contexts, limiting its use where labels are unavailable.

To achieve comprehensive and effective concept drift detection, combining data distribution-based and error rate-based methods is essential. This integration allows for detecting sudden and gradual drifts, enhancing system robustness and responsiveness [Beshah, Abebe, and Melaku, 2024].

[Jain, Kaur, and Saxena, 2022] proposed a hybrid concept drift detection method. It first uses an error rate-based approach with Bernoulli trials, then applies a data distribution-based method using Kullback-Leibler divergence to measure dissimilarity. Combining these approaches enhances robustness and accuracy in detecting concept drift by leveraging their strengths.

[Beshah, Abebe, and Melaku, 2024] proposed an ensemble combining ADWIN with the Drift Detection Method (DDM). DDM evaluates model error rate and standard deviation with predefined warning and drift thresholds to detect gradual and sudden drifts effectively. This approach leverages both techniques' strengths, enhancing the system's capability to manage diverse concept drift types.

2.2 Addressing Concept Drift in Dynamic Environments

After detecting concept drift, it's crucial to manage these changes so the learning model can adapt to new data patterns [Chuang, R.-C. Yang, and Wang, 2021]. Supervised, unsupervised, and hybrid methods enhance IDS robustness, adaptability, and accuracy in dynamic environments, ensuring continuous protection against evolving threats. The following sections describe recent works for each approach.

Supervised adaptive approaches involve training models with labeled data and updating them upon detecting concept drift [Chouchen and Jemili, 2023]. This continuous updating maintains high accuracy and reduces false positives and negatives [Seth, Singh, and Chahal, 2021]; [L. Yang and Shami, 2021].

[Seth, Singh, and Chahal, 2021] implemented an Adaptive Random Forest (ARF) with an ADWIN drift detector for IDS. ARF updates the model incrementally by integrating new samples and eliminating less informative trees, reducing the need for complete retraining. This method reduces computational and memory usage, outperforming Naive Bayes and K-nearest Neighbor (KNN) in analysis.

[L. Yang and Shami, 2021] used the Light Gradient Boosting Machine (LGBM) with Optimized Adaptive Sliding Window (OASW). LGBM, an ensemble of decision trees, excels with non-linear and high-dimensional data. Its hyperparameters are tuned with Particle Swarm Optimization (PSO). While LGBM performs well in streaming data, it lacks adaptability to evolving data without retraining.

[Chuang, R.-C. Yang, and Wang, 2021] proposed an IDS combining a real-time Adaptive XGBoost model in the fog layer with an online Random Forest model in the cloud. The real-time model detects anomalies, while the cloud model labels traffic and updates the fog layer. This approach achieves good accuracy but faces privacy risks and latency issues in IoT ecosystems.

[Wahab, 2022] implemented an online Deep Neural Network (DNN) with a PCA-based drift detector. The DNN dynamically adjusts hidden layer sizes using the Hedge weighting mechanism, improving adaptation to new data. While this approach stabilizes intrusion detection performance, it requires significant computational resources, especially on IoT devices.

[Chouchen and Jemili, 2023] proposed an ensemble approach for online intrusion detection using adaptive stream learning to handle concept drift and reduce retraining. It combines ARF and Support Vector Regression (SVR) with ADWIN. This method enhances detection precision and adapts to unknown attacks, but still requires costly labeled data for initial learning.

[Beshah, Abebe, and Melaku, 2024] introduced AUWPAE, an ensemble method for zero-day DDoS detection. It combines ARF-ADWIN, ARF-DDM, KNN-ADWIN, and Streaming Random Patches-DDM, using weighted average predictions. AUWPAE achieved high accuracy against DDoS attacks despite concept drift but increases complexity and computational demands.

[Chu et al., 2024] proposed combining XGBoost with the Whale Optimization Algorithm (WOA) for evolving data streams. XGBoost is trained on historical data,

with concept drift detected using non-parametric metrics. WOA optimizes hyper-parameters to improve performance. This approach enhances accuracy but faces challenges due to increased computational complexity and tuning time.

Unsupervised adaptive approaches enhance model adaptability and effectiveness in detecting intrusions by analyzing data without labeled examples [Xiaolan et al., 2022]. These methods identify patterns and anomalies based on data structure and distribution [Bigdeli et al., 2018]. Unsupervised techniques are useful when labeled data is scarce, improving detection by adapting to changing patterns and inherently detecting concept drift.

[Bigdeli et al., 2018] introduced an incremental anomaly detection method using Gaussian Mixture Models (GMMs). New instances are clustered and merged with existing clusters. A modified Kullback-The Leibler distance is utilized for merging based on similarity, significantly accelerating labeling and updating processes by 20 to 50 times compared to one-class SVM. However, this approach depends on the initial cluster quality and can be computationally intensive.

[Roshan et al., 2018] proposed an adaptive IDS using Extreme Learning Machines (ELM). The system clusters data with ELM's smooth approximation, identifies errors via mean squared error (MSE), and updates the model for real-time detection without full retraining. Despite its effectiveness, ELM's computational demands challenge resource-limited environments.

[Xiaolan et al., 2022] introduced the Evolving Adaptive Dynamic Network Stream Detection (EADNSD) scheme, which updates normal patterns and detects outliers through micro-clustering. It uses a global threshold for outliers and a buffer to prevent misclassification. While efficient in real-time, it assumes datasets are mostly normal, which may cause misclassification and has high memory usage.

[Zou et al., 2023] proposed an unsupervised anomaly detection algorithm that combines grid clustering with Gaussian distribution. It assigns data points to a grid,

applies filtering, and clusters grids by density and centrality, using deviations from normal clusters to compute anomaly scores. This approach is efficient for real-time applications but may struggle with high-dimensional datasets.

Hybrid approaches combine supervised and unsupervised learning to enhance accuracy, adaptability, and robustness in IDS [Jain, Kaur, and Saxena, 2022]. They detect known and unknown anomalies, improving overall capabilities. By integrating unsupervised methods, these approaches reduce dependency on extensive labeled datasets and complement the supervised learning process [Jain and Kaur, 2021].

[Jain and Kaur, 2021] proposed a distributed anomaly detection method that combines K-Means clustering with hybrid ensemble techniques, employing Random Forest (RF) and Logistic Regression as base learners, and Support Vector Machine (SVM) as the meta-learner. This approach integrates multiple algorithms to enhance detection accuracy and robustness, leveraging K-Means to reduce sample size and address data imbalance. However, the use of ensemble techniques can increase computational complexity, requiring more processing power and memory resources.

[Jain, Kaur, and Saxena, 2022] developed a network anomaly detection method combining K-means and SVM. K-means reduces sample size and labels data, while SVM classifies based on clusters. This combination enhances detection efficiency in data streams by preprocessing the data, leading to faster and more accurate classification in real-time environments. However, this approach may be sensitive to data imbalance and noise, which could impact clustering performance.

2.3 Towards Real-time Intrusion Detection: Hardware Acceleration

This section outlines the critical role of hardware acceleration in IDS. First, it reviews recent advancements aimed at enhancing system inference times to achieve real-time predictions. Next, it discusses efforts to reduce model training times, taking into

account factors such as memory usage and device resources.

Ideally, real-time intrusion detection requires predictions or inferences on new instances to occur with minimal latency, ensuring swift and effective detection [Todorov, Efnusheva, and Nikolić, 2021]. Therefore, significant research efforts have focused on optimizing inference speed through hardware acceleration, thereby enhancing performance, efficiency, and responsiveness of IDS.

[Ioannou and Fahmy, 2019] deployed an Artificial Neural Network (ANN) on FPGA SoCs for IoT gateways, enabling line-rate packet processing and adaptive model updates. They used TensorFlow for training and hardware accelerators for inference. The accelerator achieved a 161.7% performance improvement over software on the Zynq ARM core and attained up to 80.52% accuracy.

[Todorov, Efnusheva, and Nikolić, 2021] implemented a Naive Bayes (NB) classifier for IDS on a FPGA. Training was done on a CPU, and the model was imported to the FPGA for inference. The system achieved detection decisions in 240 ns, using only 2% of the FPGA area. However, the system demonstrated 70% accuracy, which may reflect Naive Bayes' sensitivity to concept drift.

[Ngo, Lightbody, et al., 2022] proposed a framework for network intrusion detection employing lightweight ANN models within IoT networks. Training is done on a GPU, while inference is accelerated by an AI microcontroller and an SoC FPGA (PYNQ-Z2). The microcontroller is 11.3 times faster than an Intel Core i7 and 21.3 times faster than an NVIDIA GPU, with the FPGA being 53.5 times faster.

Hardware acceleration significantly reduces latency during intrusion detection inference, as demonstrated by [Ioannou and Fahmy, 2019]; [Ngo, Lightbody, et al., 2022]; [Todorov, Efnusheva, and Nikolić, 2021]. However, for real-time adaptive models, minimizing training time is crucial [Roorda and Wilton, 2023]. Therefore, accelerating machine learning model training with hardware acceleration is essential.

[Maciel, Souza, and Freitas, 2019] developed a reconfigurable K-means/K-modes algorithm on FPGA for IDS. This approach allows real-time adjustments of parameters like centroids and iterations, showing up to 91% fewer cycles and 99% less energy consumption compared to parallel software on Intel Xeon processors.

[Elnawawy, Sagahyroon, and Shanableh, 2020] implemented an RF classifier on FPGA, comparing it to NB, SVM, KNN, and ANN. The FPGA-based RF classifier achieved 96.5% accuracy and a 0.834 F-score, with speedups of up to 92.64x and 47.68x on the UNIBS and UNB datasets, respectively, resulting in an average throughput of 163.24 Gbps.

[Murovič and Trost, 2020] proposed an architecture for Binary Neural Networks (BNNs) on FPGAs for edge-based IDS, emphasizing resource optimization through binary weights and activations. The results showed a 39% improvement in slice usage, a 28.2% reduction in nets used, and a 51.9% decrease in power consumption, though excessive weight minimization may compromise inference accuracy.

[Todorov, Efnusheva, and Nikolić, 2021] proposed a hardware-based Naive Bayes IDS on a Virtex 7 VC709 FPGA board, achieving a 240ns inference time with only 2% FPGA area usage. However, the accuracy was 70%, making it suitable for low-latency, low-resource applications but with lower performance.

[Zeng and Hara-Azumi, 2024] developed a RF classifier for real-time IoT IDS, utilizing an ensemble feature selection technique based on the Gini index and Out-of-Bag (OOB) score. This was implemented within a hardware/software co-design framework on a heterogeneous Zynq SoC. The Verilog HDL RF accelerator on FPGA achieved a 135x speedup over the ARM Cortex-A9, with a 1.07 ms inference time.

2.4 Discussion and Motivation

This section provides an overview of the literature, summarized in Tables 1 , 2, and 3, focusing on challenges in detecting and handling concept drift, real-time adaptive model implementation, and hardware acceleration to enhance IDS performance.

Supervised adaptive approaches effectively detect and adapt to attacks, with the ARF classifier excelling on complex datasets [Seth, Singh, and Chahal, 2021]; [Chouchen and Jemili, 2023]. Hybrid approaches, which combine K-Means with supervised learning, enhance performance by leveraging K-Means' efficiency and its capability to operate without labeled data [Jain and Kaur, 2021]; [Jain, Kaur, and Saxena, 2022]. This approach updates models only upon detecting concept drift, thereby conserving resources compared to continuous clustering [Bigdeli et al., 2018]; [Xiaolan et al., 2022]. Nevertheless, since these approaches utilize unsupervised learning for model updates, they may experience reduced performance due to data imbalance and the presence of noisy data [Jain, Kaur, and Saxena, 2022].

Error rate-based concept drift detection typically requires ground-truth labels for optimal performance [L. Yang and Shami, 2021]. However, it can be adapted to unsupervised methods by using K-Means for data labeling [Jain and Kaur, 2021]. Combining this approach with distribution-based methods and sliding window techniques enhances accuracy by capturing data changes and quantifying the severity of concept drift. This approach effectively detects both gradual and abrupt changes, even with limited labeled data [Jain, Kaur, and Saxena, 2022]. Additionally, while distribution-based methods alone cannot quantify the severity of concept drift, they can be integrated with statistical metrics to achieve this [Chu et al., 2024]; [Qahtan et al., 2015]. Finally, sliding windows enable continuous monitoring and adaptation, while adaptive windows retain extensive data patterns, thereby enhancing the model's performance and adaptability to new trends [L. Yang and Shami, 2021]; [Seth, Singh, and Chahal, 2021].

Numerous IDS studies have focused on enhancing inference speed [Todorov, Efnusheva, and Nikolić, 2021] and employing hardware acceleration for improved training efficiency [Zeng and Hara-Azumi, 2024]. However, few studies address IDS synchronization. Proper synchronization between inference and training is crucial to prevent model degradation and ensure effective intrusion detection [Roorda and Wilton, 2023]. This study proposes minimizing adaptation time through hardware-friendly algorithms, efficient architecture, and algorithmic optimization. The goal is near-simultaneous adaptation and inference to maintain real-time detection robustness without pausing the inference or prediction process.

Table 1: Overview of Adaptive Models for Intrusion Detection Systems

Reference	Method	Drift Detector	Dataset	Approach	Normalization	Feature Selection
[Roshan et al., 2018]	ELM	Drift is not detected	NSL-KDD	Unsupervised	Z-score	Not-Mentioned
[Bigdeli et al., 2018]	GMM	Drift is not detected	KDDCUP99, DARPA98, NSL-KDD, DataSetMe, and IUSTSip	Unsupervised	Not-mentioned	Not-Mentioned
[Seth, Singh, and Chahal, 2021]	ARF	ADWIN	CIC-IDS 2018	Supervised	Not-mentioned	Random Feature Selection
[L. Yang and Shami, 2021]	LGBM	OASW	IoTID20 and NSL KDD	Supervised	Not-mentioned	Not-mentioned
[Chuang, R.-C. Yang, and Wang, 2021]	Adaptive XGBoost and RF	Concept drift is not detected	UNSW-NB15 and CIC-IDS2017	Supervised	Not-mentioned	Not-mentioned
[Togbe et al., 2021]	IF	ADWIN and KSWIN	Forest Cover, Shuttle, SMTP	Unsupervised	Not-mentioned	Not-mentioned
[Jain, Kaur, and Saxena, 2022]	K-means and SVM	Kullbak-Leibler divergence and Statistical Theory	Testbed, NSL-KDD, and CIDDS-2017	Hybrid Unsupervised	Not used	Not-mentioned
[Jain and Kaur, 2021]	RF, LR, SVM and K-means	Error-based drift detection based on Acc. and FPR	Testbed, NSL-KDD, and CIDDS-2017	Hybrid Unsupervised	Min-Max and Zscore	Attribute Ratio
[Martindale, Ismail, and Talbert, 2020]	ARF and HAT	Drift Detection is controlled	KDD CUP 99	Supervised	Not-mentioned	Not-mentioned
[Abdualrahman and Ibrahem, 2021]	SGD	Drift Detection is controlled	CICIDS2017	Supervised	Z-score	C5.0 algorithm
[Yahyaoui et al., 2021]	RF	Drift is not detected	KDDCUP99 and N-BaIoT	Supervised	Not-mentioned	OneRAttributeEval
[Xiaolan et al., 2022]	EADNSD	Drift is not detected	KDDCUP99, NSL-KDD, and CIDDS–001	Unsupervised	Min-Max	Not-mentioned
[Chouchen and Jemili, 2023]	ARF and SVR	ADWIN	Fukuda laboratory Dataset	Supervised	Not-mentioned	Not-mentioned
[Wahab, 2022]	DNN	PCA	DS2OS traffic traces	Supervised	Not-mentioned	Not-mentioned
[Zou et al., 2023]	GC-ADS	ADWIN+DDM	Nab dataset	Unsupervised	Not-mentioned	Not-mentioned
[Beshah, Abebe, and Melaku, 2024]	AUWPAE	ADWIN+DDM	IoTID20 and CICIoT2023	Supervised	Z-score and Min-Max	IG and Pearson Correlation

Table 3: Overview of the State-of-the-Art IDS Utilizing Hardware Acceleration

Reference	Method	Datasets	Accuracy	Speed-up	Adaptive	Latency	Bandwidth
[Pham-Quoc, Bao, and Thinh, 2023]	Autoencoder and MLP	NSL-KDD, UNSW-NB15, CICIDS2017	90.87%, 87.49%, 98.22%	12.46x vs CPU, 31.16x vs GPU	No	Not-mentioned	8.7 Gbps, 34.74 Gbps
[Ngo, Temko, et al., 2021]	ANN	IoT-23	99.43%	6.6x vs GPU, 40.5x vs CPU	No	Not-mentioned	Not-mentioned
[Ioannou and Fahmy, 2019]	ANN	NSL-KDD	96.02%, 80.52%	161.7x vs CPU	No	0.4 μs	10 Gbps
[Ngo, Lightbody, et al., 2022]	ANN	IoT-23, UNSW-NB15	98.57%, 99.66%	11.3x vs CPU, 21.3x vs GPU	No	0.43 ms	453.5 Gbps
[Vreča et al., 2021]	BNN	UNSW-NB15	82.10%	128x vs CPU	No	Not-mentioned	Not-mentioned
[Murovič and Trost, 2021]	BNN	NSL-KDD, UNSW-NB15	77.28-98.96%	Not-mentioned	No	16 ns, 19 ns	288 Gbps, 256 Gbps
[Gordon et al., 2021]	K-NN	UNSW-NB15	95%	14.25x vs CPU	No	4 ms	Not-mentioned
[Nsunza, Tetteh, and Hei, 2018]	CNN	NSL-KDD	82.83%	Not-mentioned	No	Not-mentioned	Not-mentioned
[Murovič and Trost, 2020]	BNN	UNSW-NB15	90.74%	Not-mentioned	No	23.5 ns	Not-mentioned
[Murovic and Trost, 2019]	BNN	NSL-KDD, UNSW-NB15	77.7%, 98.96%	Not-mentioned	No	19.6 ns	Not-mentioned
[Maciel, Souza, and Freitas, 2019]	K-means and K-Modes	NSL-KDD	Not-mentioned	15x, 994x vs CPU	No	1.2 ms	Not-mentioned
[Ngo, Tran-Thanh, et al., 2019]	ANN, DT	NSL-KDD	87.30%, 95.19%	83x vs CPU, 11x vs GPU	No	12 ms, 4 ms	9.58 Gbps
[Elnawawy, Sagahyroon, and Shanableh, 2020]	RF	UNIBS, UNB	98.50%	92.64x, 47.68x vs CPU	No	28.571 ns/T	163.24 Gbps
[Ridder, Chen, and Alachiotis, 2023]	ARF	KDD CUP 99	98.42%	33.82x vs CPU, 21.21x vs GPU	Yes	19.99 s	Not-mentioned

Table 2: Performance Comparison: Accuracy, Precision, Detection Rate, False Alarm Rate, F1-Score and Latency Across Related Works

Reference	IoTID20					NSL-KDD					CICIDS2017					KDDCUP'99					UNSW-NB15				
x	Accuracy	Precision	DR	FAR	F1-Score	Accuracy	Precision	DR	FAR	F1-Score	Accuracy	Precision	DR	FAR	F1-Score	Accuracy	Precision	DR	FAR	F1-Score	Accuracy	Precision	DR	FAR	F1-score
[Seth, Singh, and Chahal, 2021]											0.9950	0.9992	0.9980		0.9242										
											Time not specified															
[L. Yang and Shami, 2021]	0.9992	0.9993	0.9998	x	0.9996	0.9831	0.9857	0.9830	x	0.9843															
	7.8ms x 62,578 instances.					9.1ms x 35,140 instances																			
[Chuang, R.-C. Yang, and Wang, 2021]											0.9922	0.9681	0.8355	x	0.8969						0.9922	0.9681	0.9277	x	0.9226
											1s x 86,253.9 instances										1s x 86,253.9 instances				
[Jain, Kaur, and Saxena, 2022]						0.9133	0.8830	0.9170	0.0211	0.8960	0.9980	0.9910	0.9920	0.0592	0.9915										
						Time is not specified					Time is not specified														
[Jain and Kaur, 2021]						0.9300	0.9600	0.9400	0.0960	0.9400	0.9800	0.9700	0.9800	0.0200	0.9800										
						66.1s x 50,000 instances					54.52s x 50,000 instances														
[Martindale, Ismail, and Talbert, 2020]																ROC Curve									
																55.59s x 700,000 instances									
[Abdualrahman and Ibrahem, 2021]											0.9944	x	x	x	x										
											0.5383 T/sec														
[Yahyaoui et al., 2021]																0.9996	x	x	x	x					
																192,307 p/s									
[Roshan et al., 2018]						x	x	0.7700	0.0305	x															
						Time is not specified																			
[Bigdeli et al., 2018]						x	x	0.8500	0.0700	x						x	x	0.9800	0.0200	x					
						245s										225s									
[Xiaolan et al., 2022]						0.9180	x	0.9010	0.0760	x	0.9886	x	0.9999	0.0125	x	0.9310	x	0.9935	0.0750	x					
						15s x 100,550 instances					15s x 100,550 instances					15s x 100,550 instances									
[Beshah, Abebe, and Melaku, 2024]	0.9954	0.9951	0.9999	x	0.9976																				
	1s x 365 instances																								

3 Objectives

This section delineates the problem statement, overarching goals, specific objectives, research questions, and anticipated contributions, providing a comprehensive framework for understanding the study's scope, aims, and expected impact.

3.1 Problem Statement

In the dynamic landscape of the IoT, the increasing proliferation of interconnected devices has heightened the need to implement robust cybersecurity measures. However, IDS face significant challenges, such as `concept drift`, which compromises their effectiveness [L. Yang and Shami, 2021]. To address this issue, it is crucial to adopt incremental and unsupervised learning algorithms in adaptive IDS, emphasizing the detection and management of concept drift. These algorithms enable effective detection of unknown attacks and handle changes in data patterns, ensuring resilience against emerging threats [Jain, Kaur, and Saxena, 2022]. However, the real-time implementation of adaptive IDS is hindered by `adaptation time` issues, as the inference process cannot be paused to perform necessary adaptations [Roorda and Wilton, 2023]. Therefore, leveraging hardware acceleration and algorithm optimization becomes indispensable for addressing these challenges, particularly those associated with real-time processing and the limited resources of IoT devices [Ridder, Chen, and Alachiotis, 2023]; [Zeng and Hara-Azumi, 2024].

3.2 Research Questions

1. What are the most effective incremental learning algorithms for adapting IDS to concept drift in real-time IoT data streams, and what are their limitations?

2. What are the most effective methods for detecting concept drift in IoT stream-

ing data, and how can the severity of this drift be quantified?

3. What are the primary challenges associated with adaptation time in adaptive models for real-time IoT IDS, and what techniques can be utilized to mitigate these challenges to enhance system effectiveness and overall performance?

4. What are the primary challenges in implementing hardware accelerators for real-time IoT IDS, and what strategies can be employed to effectively address these challenges to enhance adaptation time and concept drift detection?

3.3 General Objective

To design and implement an adaptive model with concept drift detection for Intrusion Detection Systems (IDS) to protect Internet of Things (IoT) networks. The model will leverage incremental learning on dynamic data streams, balancing inference speed with adaptation to concept drift at the algorithmic level. It will be supported by hardware acceleration techniques to enhance adaptation time, minimize performance degradation, and ensure real-time detection.

3.4 Specific Objectives

- ◇ Investigate and analyze the dynamics of data in IoT environments, identify patterns of concept drift and associated security challenges.

- ◇ Design and implement an adaptive model with concept drift detection to enhance IDS capabilities, facilitating real-time updates in dynamic data streams without dependence on labeled datasets.

- ◇ To optimize the proposed adaptive model's algorithms, including its concept drift detection, and to explore hardware acceleration techniques to reduce adaptation time, minimize model degradation, and ensure real-time detection.

- ◇ Evaluate the IDS's effectiveness and adaptability by focusing on key metrics including accuracy, precision, F1-score, detection rate, false alarm rate, and the balance between inference and adaptation time over time.

3.5 Expected Contributions

1. `Adaptive IDS Model:` Develop an adaptive model for IoT data streams to effectively respond to concept drift, improving IDS performance in dynamic environments.

2. `Concept Drift Detection Method:` Propose a concept drift detection method that identifies and measures the severity of drift to prevent model degradation over time, thereby ensuring sustained accuracy and reliability in dynamic environments.

3. `Real-time Intrusion Detection.` Optimize the adaptive model algorithmically and use hardware accelerators to minimize adaptation time while balancing inference and adaptation speeds for rapid threat detection.

4. `Rigorous Experimental Validation.` Undertake thorough experimentation in real-world IoT scenarios to quantitatively evaluate the performance of IDS. Emphasis will be placed on its capacity to manage data streams affected by concept drift and maintain resilience over time.

4 Methodology

The objective of this proposal is to develop an adaptive model with concept drift detection for real-time IDS on IoT data streams. This section outlines the overall research strategy, including the techniques and approaches to be used and the proposed extensions or modifications. It also describes the evaluation approach, encompassing

the experimental setup, simulation, and testing procedures. Figure 2 provides an overview of the adaptive IDS architecture, illustrating its key components and their interactions within the system for better understanding.

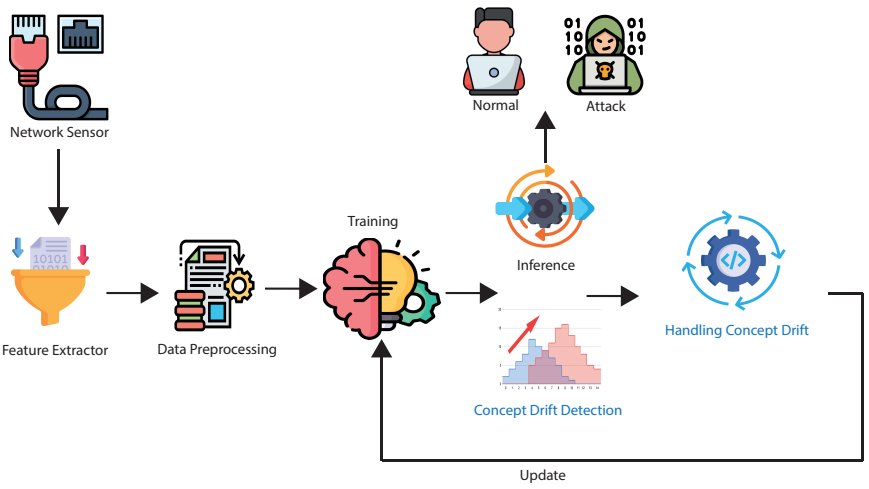


Figure 2: Overview of the Adaptive IDS Architecture, Including Its Key Components

The following will outline the research strategy, as well as the proposed extensions and modifications to the IDS components.

- **`Data Collection:`** Collect network traffic data from trusted sources, such as public databases, to avoid network sensors and feature extraction.
- **`Preprocessing:`** Implement and analyze normalization techniques like min-max and z-score normalization, and evaluate transformation methods for categorical data, such as label and one-hot encoding.
- **`Training:`** Implement and evaluate supervised algorithms, such as RF, SVM, ANN, and Naive Bayes, among others, for initial anomaly detection, leveraging their proven effectiveness in classification tasks.
- **`Development of a Concept Drift Detection Method:`** Investigate concept drift detection methods using error metrics and data distribution analysis.

- ◇ **Extensions and Modifications:** Design and implement a hybrid concept drift detection method to effectively identify and respond to both gradual and abrupt shifts in data patterns. Integrate dimensionality reduction techniques, such as PCA, with non-parametric statistical measures to quantify the extent of concept drift. Additionally, explore metrics like Detection Rate and False Alarm Rate to evaluate model error and identify potential concept drift scenarios.

- **Development of an Adaptive Model for Handling Concept Drift:** Develop and analyze adaptive models for concept drift, using clustering for pattern identification and supervised incremental learning for updates.

  - ◇ **Extensions and Modifications:** Design and implement a hybrid approach that uses both supervised and unsupervised learning to dynamically handle concept drift, activating drift handling only when needed. Implement incremental learning algorithms like ARF to minimize resource consumption, and utilize similarity metrics to enhance data relationships and mitigate the effects of noise. Finally, employ clustering methods like K-Means for labeling data and addressing the class imbalance problem.

- **Experimental Setup:**

  - ◇ The effectiveness and efficiency of various adaptive models will be evaluated using the IoTID20 [Ullah and Mahmoud, 2020] and NSL-KDD [Tavallaee et al., 2009] datasets.

  - ◇ **Evaluation Metrics for Attack Detection:**

    - ∗ `Accuracy.` This metric measures the proportion of correct classifications among total instances, as illustrated in Eq. (1).

$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP} \tag{1}$$

* `Precision.` This measure evaluates the proportion of true positives to all positives predicted by the model, as outlined in Eq. (2).

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

* `Detection Rate (DR).` This metric evaluates the proportion of correct predictions relative to missed instances, as depicted in Eq. (3).

$$DR = \frac{TP}{TP + FN} \tag{3}$$

* The `False Alarm Rate (FAR)` quantifies the proportion of negative instances misclassified as positive, as defined in Eq. (4).

$$FAR = \frac{FP}{FP + TN} \tag{4}$$

* `F1-score` combines precision and recall, balancing them in a classification model's performance, as shown in Eq. (5).

$$2 * \frac{Precision * Recall}{Precision + Recall} \tag{5}$$

- **`Simulation and Testing:`**

  ◇ **`Data Partitioning Using Sliding Windows:`** Preprocess the dataset into data streams and use a sliding window technique for incremental analysis. This approach manages large volumes, detects changes, and adapts to concept drift. Optimize the window parameters to match data characteristics and analysis needs.

  * `Extensions and Modifications:` Investigate adaptive sliding windows to capture relevant information and detect underlying structures for improved concept drift management and model performance.

  ◇ **`Performance Analysis:`** Evaluate the model's performance with and without concept drift handling using the experimental setup. Assess the method's effectiveness in detecting and measuring concept drift. Compare

training and update times with inference time, and examine their impact on model performance and degradation over time.

- ◇ **`Extensions and Modifications:`** Design and Implementation of Hardware Architecture for Enhanced Adaptive IDS. This involves exploring a range of platforms, including GPUs, FPGAs, and homogeneous hardware systems, as well as examining various code-design approaches. The aim is to thoroughly assess these technologies and identify the most effective solution for addressing model degradation over time.

4.1 Justification of the Methodology

The presented methodology addresses the need to effectively manage concept drift in IoT environments where resources and time are constrained. Many machine learning algorithms can handle concept drift, but some are slow, complex, or memory-intensive, making them unsuitable for real-time applications. Additionally, some methods rely on labeled datasets, which may not be available. Traditional fixed sliding windows can be inefficient, as they may either miss critical information or include irrelevant data, leading to suboptimal performance. To overcome these limitations, adaptive sliding windows are essential for capturing the most relevant information dynamically. Hybrid unsupervised approaches with incremental learning can be used, managing concept drift only when necessary and optimizing processing time and memory usage. Effective management also requires detecting various types of concept drift and measuring their severity to fine-tune models accordingly. Furthermore, incorporating hardware acceleration significantly enhances training and adjustment speeds, preventing model degradation and ensuring robust performance in dynamic settings. Thus, the methodology's integration of hybrid approaches, comprehensive drift detection, adaptive sliding windows, and hardware acceleration provides a well-rounded solution for real-time IoT scenarios.

5 Preliminary Results

This section details the proposed adaptive model, which incorporates a concept drift detection method. These advancements align with the components of `CONCEPT DRIFT DETECTION` and `HANDLING CONCEPT DRIFT` outlined in Section 4, which are integral to the proposed methodology. Furthermore, the experimental analysis is elaborated upon, and finally, a comprehensive discussion of the results is presented.

5.1 Proposal for Detecting and Handling Concept Drift

The proposed approach for detecting and managing concept drift integrates two types of sliding windows to handle dynamic data: a `fixed sliding window` for stability in model performance and an `adaptive sliding window` for retraining to accommodate new data patterns. This adaptive mechanism enables the system to continually adapt and refine its understanding, ensuring robustness and accuracy in handling evolving data patterns. The concept drift detection is founded upon the analysis of data distribution and error rate analysis, with a hybrid unsupervised approach being employed for managing the concept drift. Subsequent subsections detail the proposed adaptive model with concept drift detection.

5.1.1 Proposed Concept Drift Detection Method

The proposed method for concept drift detection harnesses PCA alongside Spearman's Correlation Coefficient (SCC). In data streams without drifts, the monotonic relationship is strong. The explanation is simple: during an attack, certain features such as connection time, energy consumption, and traffic level consistently rise on the targeted device. However, in a firmware update, while values like connection time, energy consumption, and traffic levels may rise, irregular patterns do not necessarily follow suit. This could suggest the presence of concept drift between two

consecutive fixed sliding windows.

`SCC` is a statistical measure that evaluates the strength and direction of the relationship between two ordinal or quantitative variables [Ali Abd Al-Hameed, 2022]. Unlike Pearson's correlation coefficient, which assesses the linear relationship between variables, Spearman evaluates the relationship based on the ranks of variable values. This feature allows it to capture relationships that do not necessarily follow a *linear pattern*. The formula for SCC is given by Eq. 6.

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)} \tag{6}$$

Where ρ is the SCC, d_i are the differences between the ranks of the two variables, and n is the number of observations. `PCA` is a statistical technique used to reduce the dimensionality of a dataset while preserving most of its variability [Greenacre et al., 2022]. It achieves this by transforming the original variables into a new set of variables, called principal components, which are linear combinations of the original variables. These principal components are ordered in terms of the amount of variability they explain in the data, with the first component explaining the most variability, followed by the second, and so on.

In the proposed approach, PCA is applied to each fixed sliding window SW_i to extract the two principal components PC_1 and PC_2. These components represent the most significant variance in the data. SCC is then used to measure the monotonic relationship between these two components. For each sliding window SW_i, PCA is performed to reduce the dimensionality of the data to the two principal components PC_1 and PC_2. SCC (ρ_i) is computed between PC_1 and PC_2 within SW_i. SCC evaluates the monotonic relationship, capturing both linear and nonlinear correlations. In data streams without changes in the underlying concept, SCC is expected to remain constant across different sliding windows (SW_i, SW_{i+1}). A significant change δ in SCC between (ρ_1) and (ρ_2) indicates potential concept drift.

A large δ may suggest an abrupt change, whereas a small δ could represent a gradual change. Therefore, (δ) can serve as an indicator of the severity of the concept drift.

Algorithm 1 Concept Drift Detection Method

1: Input: Sliding windows SW_i, SW_{i+1}, Threshold $threshold$
2: Output: Concept drift detection (Boolean)
3: procedure PCA(SW_i, SW_{i+1})
4: $pcs_i \leftarrow PCA(SW_i)$
5: $pcs_{i+1} \leftarrow PCA(SW_{i+1})$
6: $d_1 \leftarrow pcs_i[1]$
7: $d_2 \leftarrow pcs_i[2]$
8: $d_3 \leftarrow pcs_{i+1}[1]$
9: $d_4 \leftarrow pcs_{i+1}[2]$
10: return (d_1, d_2, d_3, d_4)
11: procedure Spearman_Calculation($(pc1_i, pc2_i, pc1_{i+1}, pc2_{i+1})$)
12: $\rho_1 \leftarrow SpearmanCorr(d_1, d_2)$
13: $\rho_2 \leftarrow SpearmanCorr(d_3, d_4)$
14: return (ρ_1, ρ_2)
15: procedure Drift_Detection(SW_i, SW_{i+1}, $threshold$)
16: $(d_1, d_2, d_3, d_4) \leftarrow PCA(SW_i, SW_{i+1})$
17: $(\rho_1, \rho_2) \leftarrow Spearman\_Calculation(d_1, d_2, d_3, d_4)$
18: if $(\delta \leftarrow (\rho_1 - \rho_2) <$ threshold) then
19: Concept drift
20: else
21: Concept drift not detected

Algorithm 1 continuously monitors the data stream, applies PCA for dimensionality reduction (see lines 3-10 in Algoritm 1), and uses SCC to evaluate the monotonic relationship between the principal components (see lines 11-14 in Algorithm 1). A significant change in SCC between consecutive windows indicates concept drift (see lines 18-19 in Algoritm 1). For practical implementation, selecting the sliding window size and the threshold δ for detecting significant changes in SCC is crucial. These parameters should be optimized based on the specific characteristics of the data stream and the application requirements.

5.1.2 Enhancing Drift Handling with K-Means and Random Forest

To address concept drift, the adaptability of K-Means and RF to evolving data patterns over time is leveraged. A comprehensive strategy for enhancing drift handling is outlined in Algorithm 2, providing a detailed process. Firstly, the data are divided into fixed sliding windows of size n, forming a set $S = (SW_1, SW_2, SW_3, ..., SW_n)$. Subsequently, Algorithm 1 is applied to precisely detect concept drift between two sliding windows (SW_i, SW_{i+1}). To effectively manage potential concept drift propagation to consecutive windows, an adaptive sliding window is utilized to retain these windows and extract the necessary information (see lines 3-10 in Algorithm 2).

The proposed approach suggests using the adaptive sliding window to identify clusters with similar statistical distributions. To achieve this, a pairwise similarity matrix is constructed using the Manhattan distance (see line 22 in Algorithm 2). This metric is selected for its strong generalization to higher dimensions, efficiency in parallel processing environments, and reduced computation time [Pandit, Gupta, et al., 2011], all of which are crucial for real-time adaptive IDS models.

K-Means is used to partition the pairwise distance matrix into k clusters, with k being a user-defined parameter (see line 23 in Algorithm 2). This strategy ensures that the clustering process not only organizes the data into meaningful clusters but also captures the inherent relationships between instances, leasing to more accurate and insightful results [Geng and Tang, 2020]. To determine the optimal value for k, Canopy clustering is employed. This method evaluates various potential values for k and suggests initial cluster centers [Jain, Kaur, and Saxena, 2022].

The proposed model addresses two scenarios related to concept drift: (1) instances previously classified as *normal* are now *attack*, reducing the Detection Rate (DR) and indicating new threats; (2) instances previously *attack* are now *normal*, increasing the FAR and suggesting changes in network behavior. To effectively define these scenarios, a concept drift detection method based on error rates, specifically

focusing on DR and FAR, is proposed as outlined in Algorithm 2 (see lines 11-18 in Algorithm 2). Given the absence of ground-truth labels, clusters generated by K-Means are utilized to assign labels [Jain, Kaur, and Saxena, 2022]; [Jain and Kaur, 2021]. This approach facilitates the effective determination of each scenario type, allowing for a robust identification and response to evolving data patterns.

If the DR metric decreases, it is crucial to identify *attack* instances that the current model fails to recognize. To address this, clusters with the least variance are selected to focus on significant examples according to their similarity, reducing sample size and managing data imbalance. Previous research shows that clustering enhances model performance by focusing on representative examples, effectively addressing data imbalance [Jain and Kaur, 2021]; [Jain, Kaur, and Saxena, 2022]. Analyzing these low-variance clusters helps differentiate between correctly and incorrectly classified distributions. The Random Forest classifier is used to classify these low-variance clusters into normal and attack categories. When concept drift occurs, it reveals new attacks that may be misclassified as normal, allowing for model refinement (see lines 26-28 in Algorithm 2). When the FAR increases, previously classified *attack* instances are reassessed to enhance the classification of *normal* instances (see lines 29-31 in Algorithm 2). Fig. 3 illustrates how the model handles concept drift in both scenarios.

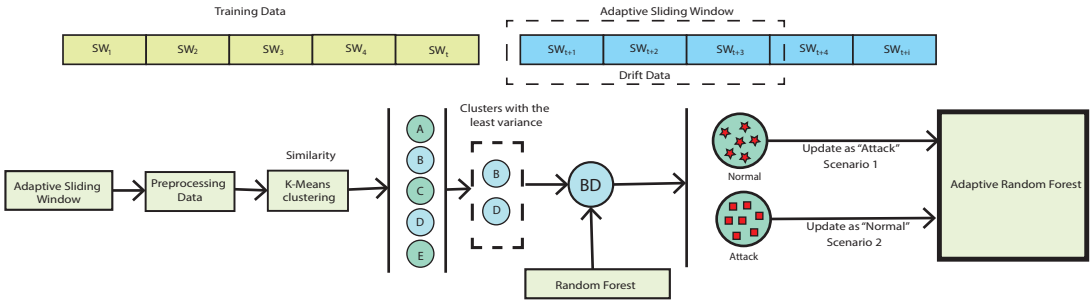


Figure 3: Hybrid Approach: K-Means & RF for Effective Concept Drift Handling

Algorithm 2 Concept Drift Manager

Input: Testing Data, Threshold: $Thresh\_DR$, Threshold: $Thresh\_FAR$
Output: Update Adaptive Random Forest Model
procedure Adaptive_Sliding_window(Testing data)
    Divide the testing data into fixed sliding windows of size n, forming a set $S = \{SW_1, SW_2, SW_3, ..., SW_n\}$.
    for each sliding window SW_i in the set S do
        if Drift_Detection(SW_i, SW_{i+1}) == True then ▷ Algorithm 1
            Add SW_{i+1} to the Adaptive Sliding Window (ASW).
        else
            Predict on SW_{i+1}
    return ASW
procedure scenario_identification(Thresh_DR, Thresh_FAR)
    Perform Random Forest classifier on SW_i
    $(DR, FAR) \leftarrow PredictWithCurrentModel(SW_{i+1})$
    if DR < Thresh_DR then
        $scenario \leftarrow 1$
    else if FAR > Thresh_FAR then
        $scenario \leftarrow 2$
    return $scenario$
procedure Handling_concept_drift(Testing_Data, Thresh_DR, Thresh_FAR)
    $scenario \leftarrow SCENARIO\_IDENTIFICATION(Thresh\_DR, Thresh\_FAR)$
    $ASW \leftarrow ADAPTIVE\_SLIDING\_WINDOW(Testing\_Data)$
    $X \leftarrow PairwiseDistanceCalculation(ASW)$
    $(c_x = \{c_1, c_2, c_3, c_4, c_5\}) \leftarrow KMeans(X, k = 5)$
    $k_x \leftarrow Lowest\_Variance(c_x)$
    $(normal, attacks) \leftarrow PredictWithCurrentModel(k_x)$
    if $scenario == 1$ then
        $new\_attack\_instances \leftarrow normal$
        $current\_model \leftarrow UpdateRandomForestModel(new\_attack\_instances)$

    else if $scenario == 2$ then
        $new\_normal\_instances \leftarrow attacks$
        $current\_model \leftarrow UpdateRandomForestModel(new\_normal\_instances)$

5.1.3 Adaptive Random Forest Training and Prediction

The primary aim of this study is to detect anomalies in network traffic. For this purpose, an ARF classifier was chosen due to its effectiveness in detecting and adapting to attacks [Leon, Markovic, and Punnekkat, 2022]; [Seth, Singh, and Chahal, 2021]. Due to the presence of concept drift, a model trained once cannot be reliably applied at all times, necessitating retraining upon detecting drift. To ascertain drift in traffic, an SCC between the principal components of consecutive sliding windows is calculated over time. Fig. 4 provides an overview of the proposed approach for managing concept drift and detecting anomalies. The next section provides the results obtained using the proposed adaptive model.

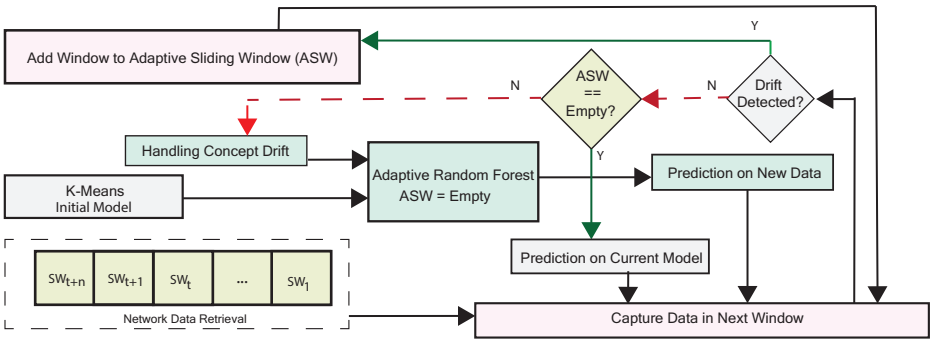


Figure 4: Overview of the Proposed Adaptive IDS with Concept Drift Detection

In this section, an adaptive model with concept drift detection for IDS is presented to address the challenges posed by concept drift. It explains how this approach enables the system to detect concept drift and dynamically adapt to fluctuations in input data, thereby enhancing its intrusion detection capabilities. In the following section, a detailed analysis of the obtained results will be conducted, evaluating the system's performance according to the methodology outlined in this document.

5.2 Experimental Analysis

The primary aim of the experimental results is to evaluate the adaptive model's ability to adjust to evolving data when detecting concept drift. These experiments were conducted on Google Colab, utilizing hardware resources including an Intel(R) Xeon(R) CPU @ 2.20GHz and 12.67 GB of RAM.

5.2.1 Analysis of the NSL-KDD dataset

The `NSL-KDD dataset` is a commonly preferred public dataset for network intrusion detection research [Jain and Kaur, 2021]. It contains 148,517 records with 41 features plus a label class as the 42nd feature. Among the features, 32 are continuous, and 9 are categorical. The dataset is divided into training and test sets. The training set includes 22 attack types, categorized into four classes: `Denial of Service (DoS)`, `User to Root (U2R)`, `Remote to Local (R2L)`, and `Probe`. The test set includes 17 additional attack types in the same classes. Table 4 illustrates the attack types in both sets; bold text denotes new attacks in the test set, while italic text indicates those exclusive to the training set. A notable drift from the training set to the test set occurs due to the absence of certain unknown attacks in the training set, leading to their misclassification as normal traffic in the test set [L. Yang and Shami, 2021].

Table 4: NSL-KDD Dataset Attack Types

DoS	Probe	R2L	U2R
apache2, back, land, mailbomb, neptune, pod, processtable, smurf, teardrop, udpstorm	ipsweep, mscan, nmap, portsweep, saint, satan	spy, warezclient, ftp_write, guesspasswd, httptunnel, imap, multihop, named, phf, sendmail, snmpgetattack, warezmaster, xlock, xsnopp	bufferoverflow, loadmodule, perl, ps, rootkit, snmpguess, sqlattack, worm, xterm

Table 5: Concept Drift Indicator Between SW_0 and Subsequent Sliding Windows

SW_0	SW_1	SW_2	SW_3	SW_4	SW_5	SW_6	SW_7	SW_8	SW_9	SW_{10}
δ	0.0127	0.0584	0.0073	0.2550	0.2871	0.0274	0.0122	0.0059	0.2333	0.0284

Firstly, categorical attributes were converted to numerical values using `label encoding`, and feature values were normalized with `min-max normalization` to ensure equal contribution and improve convergence during training. This preprocessing step is crucial for maintaining data integrity and enhancing the adaptive IDS's effectiveness. For data integration, two sets were merged by randomly selecting windows from each. A downsized NSL-KDD dataset with 5.0×10^3 instances was selected after testing various sliding window sizes to determine the optimal size. It is essential to balance detection sensitivity and model performance since larger windows can dilute concept drifts, while smaller windows might reduce effective detection.

After randomly merging the windows from the two sets, the objective is to detect concept drift between the sliding windows. This involves monitoring changes in the data distribution over time to identify shifts in patterns or relationships. Table 5 presents the results obtained after employing the proposed concept drift detection method described in Subsection 5.1.1. As observed in Table 5, there are no significant changes (δ) in the SCC across sliding windows SW_0 through SW_3. However, this pattern is disrupted between windows SW_3 and SW_4, where a sudden change becomes noticeable. This suggests a significant abrupt conceptual change, which persists in the subsequent window SW_5, and finally reappears in window SW_9. In this case, the adaptive sliding window is initially filled with data from windows SW_4 and SW_5 and is then used to update the model. Additionally, as shown in Table 5, the value of (δ) in SW_9 is lower than in SW_4 and SW_5, which is reflected in the accuracy results presented in Table 6. This demonstrates the method's effectiveness in detecting gradual changes.

Figure 5 depicts the data distribution of the two principal components within

sliding windows SW_3 and SW_4, along with the SCC for each of them. In SW_3, a consistent pattern is evident, as the points tend to follow a uniform trajectory in the graph, indicating a coherent relationship between the variables represented by the principal components. However, in the presence of a concept drift in SW_4, the monotonic relationship between the points diminishes. This suggests that the points may scatter more randomly on the graph or follow a less predictable trajectory.

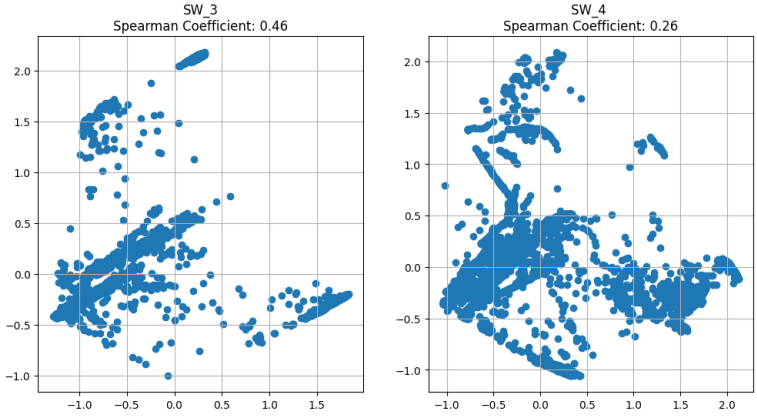


Figure 5: Data Distribution of Principal Components and SSC of SW_3 and SW_4

Tables 6 and 7 present the accuracy, precision, DR, FAR, and F1-score before and after drift handling in the NSL-KDD dataset. Table 6 highlights a significant decline in anomaly detection performance for (SW_3, SW_4), (SW_4, SW_5), and (SW_8, SW_9). This highlights the effectiveness of the proposed method in identifying abrupt concept drift through distribution changes. Consequently, prediction of the subsequent windowed data was based on the newly adapted model. Enhanced accuracy, precision, DR, FAR, and F1-score are evident in Table 7. These improvements observed further validate the system's capability in mitigating drift-induced performance degradation. Figure 6a illustrates the evolution of accuracy over time. The graph shows accuracy fluctuations over time, with notable increases at specific intervals. For instance, a significant rise in accuracy occurs between the fourth and fifth data points, attributed to concept drift management.

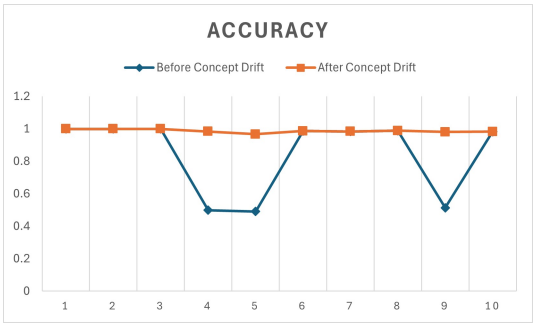
Figure 6b illustrates that the adaptation time is considerably higher for 1×10^4

Table 6: Performance Evaluation Before Drift Handling (NSL-KDD dataset)

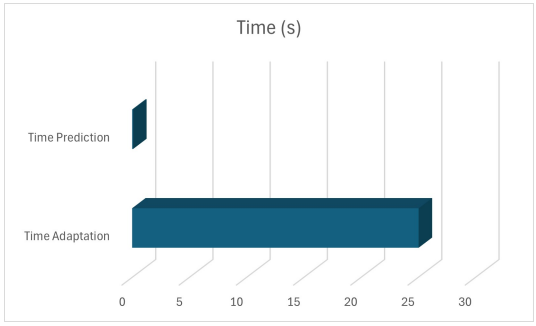
Sliding Window Id	Accuracy	Precision	DR	FAR	F1-score	Drift Detected
(SW_0, SW_1)	0.9988	0.9985	0.9985	0.001	0.9985	No
(SW_1, SW_2)	0.9990	0.9995	0.9980	0.0003	0.9987	No
(SW_2, SW_3)	0.9990	1	0.9974	0	0.9987	No
(SW_3, SW_4)	0.4986	0.9596	0.4021	0.0748	0.5667	Yes
(SW_4, SW_5)	0.4904	0.9623	0.3930	0.0695	0.5580	Yes
(SW_5, SW_6)	0.9865	0.9953	0.9715	0.0031	0.9833	No
(SW_6, SW_7)	0.9832	0.9990	0.9608	0.00006	0.9795	No
(SW_7, SW_8)	0.9884	0.9979	0.9732	0.0013	0.9854	No
(SW_8, SW_9)	0.5136	0.9609	0.4216	0.0764	0.5861	Yes
(SW_9, SW_{10})	0.9820	0.9924	0.9310	0.0050	0.9775	No
Average	0.8439	0.9865	0.8047	0.0231	0.8632	—

Table 7: Performance Evaluation After Drift Handling (NSL-KDD Dataset)

Sliding Window Id	Accuracy	Precision	DR	FAR	F1-score	Adaptation Time	Inference Time
(SW_3, SW_4)	0.9832	0.9918	0.9874	0.0357	0.9896	25.04s	0.0590s
(SW_4, SW_5)	0.9662	0.9863	0.9721	0.0607	0.9792		
(SW_8, SW_9)	0.9806	0.9913	0.9848	0.0382	0.9880	14.41s	0.0863s



(a) Accuracy of the classifier before and after applying concept drift detection technique within drift windows on the NSL-KDD dataset.



(b) Prediction Time for 5×10^3 Instances versus Adaptation Time for 1×10^4 Instances.

Figure 6: Comparison of Classifier Accuracy and Prediction vs. Adaptation Time

instances compared to the prediction time for 5×10^3 instances. From this, it is

possible to infer that in real-time scenarios, the system could process approximately 166.885 windows before the model is able to adapt. Consequently, there would be a degradation in the model's performance, as demonstrated in Table 6.

Table 8: Performance of Proposed Approach on NSL-KDD dataset

Reference	Method	Av. Accuracy	Av. Precision	Av. DR	Av. F1-Score	Av. FAR	Approach
[L. Yang and Shami, 2021]	LGBM	98.31%	98.57%	98.30%	98.43%	Not-reported	Supervised
[Roshan et al., 2018]	ELM	Not-reported	Not-reported	77%	Not-reported	3.05%	Hybrid Unsupervised
[Jain and Kaur, 2021]	RF, LR, SVM and K-means	93%	96%	94%	94%	9.60%	Hybrid Unsupervised
[Jain, Kaur, and Saxena, 2022]	K-means+SVM	91.33%	88.3%	91.7%	89.6%	2.11%	Hybrid Unsupervised
[Xiaolan et al., 2022]	EADNSD	91.80%	Not-reported	90.10%	Not-reported	7.60%	Unsupervised
[Bigdeli et al., 2018]	Incremental GMM-based clustering	Not-reported	Not-reported	85%	Not-reported	7%	Unsupervised
Proposed Approach	K-means+Random Forest	98.66%	99.52%	97.74%	99.78%	1.14%	Hybrid Unsupervised

Table 8 compares the proposed adaptive model with alternative approaches from related studies. The data shows that the proposed model generally outperforms its peers in key attack detection metrics and, in some cases, matches the performance of supervised adaptive models. This highlights the effectiveness and competitive edge of the proposed adaptive model.

5.2.2 Analysis of the IoTID20 dataset

The `IoTID20 dataset` [Ullah and Mahmoud, 2020] is a recently developed and comprehensive resource specifically designed for detecting anomalous activities in IoT networks. It includes both intrusion and regular activities recorded from devices such as Wi-Fi routers, SKT NGU computers, and EZVIZ cameras. The dataset consists of 625,783 instances and 83 attributes, with nominal attributes omitted, resulting in 79 characteristics. It provides detailed labels for intrusion detection, including binary, category, and sub-category labels. Table 9 presents the detailed distribution of dataset records between standard and intrusion operations. It also includes the binary, category, and sub-category labels found in the IoTID20 dataset.

Categorical attributes were converted to numerical values using `label encoding`, and feature values were normalized using `min-max normalization` to ensure equal

Table 9: IoTID20 Distribution

		Binary		Category		Sub_Category		75%		25%	
Dataset IoTID20	625,783	Anomaly	585,710	Anomaly-Scan	75,265	Hot Port	22,192	For Training	16,644	For Test	5,548
						Port OS	53,073	For Training	39,805	For Test	13,268
				Anomaly-Mirai	415,676	ACK Flooding	55,124	For Training	41,343	For Test	13,781
						Host BruteForceg	121,181	For Training	90,886	For Test	30,295
						HTTP Flooding	55,818	For Training	41,864	For Test	13,995
						UDP Flooding	183,553	For Training	137,665	For Test	45,888
				Anomaly-MITM ARP	35,377	MITM ARP Spoofing	35,377	For Training	26,533	For Test	8,844
				Anomaly-DOS	59,392	Synflooding	59,391	For Training	44,544	For Test	14,848
		Normally	40,073	Normal	40,073	Normal	40,073	For Training	30,055	For Test	10,018
								Total Sample	469,337	For Test	156,446

contribution and improve convergence during training. A downsized IoTID20 dataset with 5.0×10^3 instances was selected to test the detection and handling of concept drift. Table 10 presents the SCC calculated between the initial window SW_0 and the subsequent sliding windows. As illustrated in the pairs (SW_0, SW_6), (SW_0, SW_3), (SW_0, SW_7), and (SW_0, SW_4), a gradual concept drift is observed, which impacts the performance of the IDS, as detailed in Table 11. In this case, the adaptive sliding window is initially filled with data from windows SW_3 and SW_4 and is then used to update the model. The same process is applied for windows SW_6 and SW_7. Table 12 displays the performance results of the ARF classifier following concept drift handling. Table 12 highlights the improvements in accuracy, precision, DR, FAR, and F1-score achieved by the system. These enhancements demonstrate the system's effectiveness in adapting to gradual changes. However, the columns for *Adaptation Time* and *Inference Time* in Table 12 reveal that the system can process approximately 165 windows before the model is fully adapted. This suggests a trade-off between adaptation speed and real-time processing capability, underscoring the need for efficient adaptation strategies in dynamic environments.

Table 10: Concept Drift Indicator Between SW_0 and Subsequent Sliding Windows

SW_0	SW_1	SW_2	SW_3	SW_4	SW_5	SW_6	SW_7	SW_8	SW_9	SW_{10}
(δ)	-0.0034	-0.0044	-0.0126	0.0250	-0.0010	0.0105	-0.0234	0.0015	0.0027	-0.0053

Table 11: Performance Evaluation before Drift Handling (IoTID20 dataset).

Sliding Window Id	Accuracy	Precision	DR	FAR	F1-score	Drift Detected
(SW_0, SW_1)	0.9960	0.9963	0.9993	0.0568	0.9978	No
(SW_1, SW_2)	0.9960	0.9960	0.9996	0.0546	0.9978	No
(SW_2, SW_3)	0.9917	0.9932	0.9978	0.0916	0.9956	Yes
(SW_3, SW_4)	0.9940	0.9951	0.9984	0.0740	0.9968	Yes
(SW_4, SW_5)	0.9960	0.9963	0.9993	0.0558	0.9978	No
(SW_5, SW_6)	0.9951	0.9957	0.9990	0.0616	0.9974	Yes
(SW_6, SW_7)	0.9934	0.9935	0.9993	0.0853	0.9964	Yes
(SW_7, SW_8)	0.9940	0.9938	0.9996	0.0796	0.9967	No
(SW_8, SW_9)	0.9942	0.9945	0.9993	0.0841	0.9969	No
(SW_9, SW_{10})	0.9951	0.9948	1	0.0765	0.9974	No
Average	0.9945	0.9949	0.9991	0.07199	0.9970	—

Table 12: Performance Evaluation after Drift Handling (IoTID20 dataset).

Sliding Window Id	Accuracy	Precision	DR	FAR	F1-score	Adaptation Time	Inference Time
(SW_2, SW_3)	0.9971	0.9969	0.9947	0.0416	0.9984	19.24s	0.1162s
(SW_3, SW_4)	0.9988	0.9987	0.9954	0.0185	0.9993		
(SW_5, SW_6)	0.9988	0.9987	0.9981	0.0176	0.9993	18.14s	0.144s
(SW_6, SW_7)	0.9977	0.9975	0.9975	0.0325	0.9987		

Table 13: Performance of Proposed Approach on IoTID20 dataset.

Reference	Method	Av. Accuracy	Av. Precision	Av. DR	Av. F1-Score	Av. FAR	Approach
[L. Yang and Shami, 2021]	LGBM	99.92%	99.93%	99.98%	0.9996	-	Supervised
Proposed Approach	K-means+Random Forest	99.69%	99.70%	99.75%	99.83%	4.32%	Hybrid Unsupervised

Table 13 provides a comparison between the proposed adaptive model and alternative methods, such as the one suggested by [L. Yang and Shami, 2021], using the IoTID20 dataset. The results demonstrate that the performance of the proposed model is comparable to that of supervised adaptive approaches, despite not utilizing label knowledge. Additionally, these results reveal a low FAR of just 4.32%, showcasing the model's effectiveness in reducing false positives while preserving high detection accuracy.

5.3 Discussion

Section 2 emphasizes the challenges faced by adaptive models for IDS, particularly in relying on supervised learning and labeled datasets for attack detection. The proposed adaptive model seeks to address this issue by exhibiting robust detection capabilities that are comparable to those of established supervised approaches, such as those described by [L. Yang and Shami, 2021]. Unlike unsupervised approaches, such as those proposed by [Roshan et al., 2018], [Xiaolan et al., 2022] and [Bigdeli et al., 2018], which continuously cluster real-time data, the proposed adaptive model identifies and manages concept drift only when necessary. This approach optimizes processing time, resource usage, and adaptability. Hybrid approaches, such as those introduced by [Jain, Kaur, and Saxena, 2022] and [Jain and Kaur, 2021], integrate supervised and unsupervised methods to enhance detection performance while maintaining a low FAR. In contrast, the proposed adaptive model focus on identifying patterns through clustering on the similarity matrix rather than directly analyzing raw data. This methodology facilitates the grouping of data distributions based on their similarities, allowing the detection of distributions familiar to the current model using Random Forest. By applying a concept drift detection method based on DR and FAR, it becomes possible to identify distributions that the model fails to recognize, enabling timely model updates and enhancing overall performance.

Moreover, employing an adaptive window helps retain critical information and manage the propagation of concept drift, enhancing pattern recognition and facilitating model updates. The proposed concept drift method, based on distribution data, accurately identifies both abrupt and gradual changes by dynamically adjusting the threshold (δ). In contrast, traditional methods like ADWIN and PCA+Kullback-Leibler, as proposed by [Qahtan et al., 2015], are limited to detecting only abrupt changes and may miss subtler, gradual shifts.

Finally, experimental results demonstrated that the adaptation time exceeds

the inference time, highlighting the necessity to explore hardware acceleration techniques and algorithmic optimization. These measures are essential to prevent model degradation when implementing the proposed adaptive model in real-time IDS.

6 Work Plan

This section delineates the proposed work plan for the research. Figure 7 illustrates the Gantt chart detailing the timelines for each activity. The plan encompasses essential phases such as literature review, data collection, analysis, experimentation, and presentation of findings, ensuring systematic progress and optimal resource allocation. Each phase is meticulously designed to achieve the research objectives within the specified timeframes. The forthcoming section provides a detailed description of the proposed activities.

1. **Literature Review**
    (a) Review and summarize at least 50 relevant articles on the research topic.
    (b) Identify at least 3 emerging trends of interest from the literature review.
    (c) Identify at least 3 key research questions from the literature review.
2. **Objective Definition**
    (a) Identify and describe the thesis research problem concisely.
    (b) Define research objectives clearly to guide the study effectively.
    (c) Formulate at least three specific and relevant research questions.
3. **Establish Experimental Setup**
    (a) Identify at least 2 relevant datasets aligned with research objectives.
    (b) Identify at least 3 metrics aligned with study objectives.

4. **Algorithm Selection**

   (a) Evaluate at least three algorithms for handling concept drift.

   (b) Evaluate at least three concept drift methods for detection effectiveness.

5. **To Design and Implement of Adaptive Model**

   (a) Develop an adaptive model for IoT data streams with over 90% DR and under 10% FAR.

   (b) Develop a robust concept drift detection method proficient in accurately identifying shifts within IoT data streams, targeting an accuracy of 90%.

   (c) Optimize the adaptive model at the algorithmic level to achieve at least a 20% increase in processing speed and a 15% reduction in resource consumption.

6. **Experimental Analysis**

   (a) Conduct experiments to evaluate and compare the adaptive model across different concept drift scenarios using the proposed experimental setup, and benchmark it against findings from the literature.

   (b) Evaluate the limitations of the proposed adaptive model and explore areas for improvement and potential future directions.

7. **Article Writting**

   (a) Write a draft of the scientific article based on the experimental results.

   (b) Submit the article to the selected journal or conference for peer review.

8. **Proposal Doctoral Presentation**

   (a) Present and defend the doctoral proposal successfully.

9. **Design of Hardware Acceleration and Optimization of Algorithms**

(a) Design and implement a hardware architecture to accelerate the adaptive model, targeting a 50% improvement in processing speed compared to the initial software version.

10. **Experimental Analysis**

    (a) Assess acceleration hardware's impact on model performance, aiming for a 2x speedup compared to baseline without hardware.

    (b) Conduct real-time hardware tests for at least 100 Mbps throughput and 10 ms latency.

    (c) Compare results with literature to identify at least 2 contributions.

    (d) **Article Writing (Repeat 7).**

11. **Thesis Writing**

    (a) Draft and refine the entire doctoral thesis, incorporating research findings and conclusions into a cohesive document.

12. **Thesis Defense**

    (a) Present and defend the doctoral thesis successfully.

References

Abdualrahman, Amer Abdulmajeed and Mahmood Khalel Ibrahem (2021). "Intrusion Detection System Using Data Stream Classification". In: Iraqi Journal of Science, pp. 319–328.

Ali Abd Al-Hameed, Khawla (2022). "Spearman's correlation coefficient in statistical analysis". In: International Journal of Nonlinear Analysis and Applications 13.1, pp. 3249–3255.

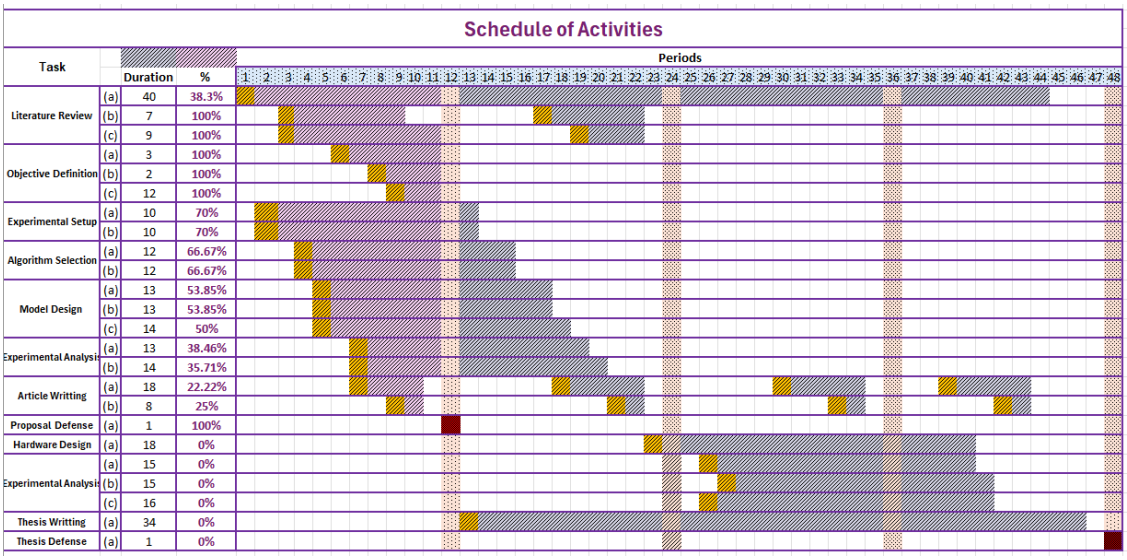


Figure 7: Proposed Activity Schedule: Gantt Diagram.

Babüroğlu, Elif Selen, Alptekin Durmuşoğlu, and Türkay Dereli (2023). "Concept drift from 1980 to 2020: a comprehensive bibliometric analysis with future research insight". In: Evolving Systems, pp. 1–21.

Beshah, Yonas Kibret, Surafel Lemma Abebe, and Henock Mulugeta Melaku (2024). "Drift Adaptive Online DDoS Attack Detection Framework for IoT System". In: Electronics 13.6, p. 1004.

Bigdeli, Elnaz et al. (2018). "Incremental anomaly detection using two-layer cluster-based structure". In: Information Sciences 429, pp. 315–331.

Chouchen, Islem and Farah Jemili (2023). "Intrusion Detection based on Incremental Learning". In: 2023 International Conference on Cyberworlds (CW). IEEE, pp. 448–455.

Chu, Renjie et al. (2024). "Intrusion detection in the IoT data streams using concept drift localization". In: AIMS Mathematics 9.1, pp. 1535–1561.

Chuang, Shih-Hsien, Ren-Chieh Yang, and Sheng-De Wang (2021). "Network intrusion detection system with stream machine learning in fog layer and online labeling in cloud layer". In: 2021 International Conference on Electronic Communications, Internet of Things and Big Data (ICEIB). IEEE, pp. 53–59.

Elnawawy, Mohammed, Assim Sagahyroon, and Tamer Shanableh (2020). “FPGA-based network traffic classification using machine learning”. In: IEEE Access 8, pp. 175637–175650.

García, María and Carlos López (2024). “Challenges in Intrusion Detection Systems Leveraging Machine Learning and Batch Processing”. In: Journal of Cybersecurity Research 12.2, pp. 87–102.

Geng, Xiurui and Hairong Tang (2020). “Clustering by connection center evolution”. In: Pattern Recognition 98, p. 107063.

Gómez, Ana and Javier Rodríguez (2024). “Challenges in Machine Learning-Based Intrusion Detection Systems for IoT”. In: Journal of Cybersecurity Research 15.1, pp. 45–58.

Gordon, Holden et al. (2021). “An efficient SDN architecture for smart home security accelerated by FPGA”. In: 2021 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN). IEEE, pp. 1–3.

Greenacre, Michael et al. (2022). “Principal component analysis”. In: Nature Reviews Methods Primers 2.1, p. 100.

Ioannou, Lenos and Suhaib A Fahmy (2019). “Network intrusion detection using neural networks on FPGA SoCs”. In: 2019 29th International Conference on Field Programmable Logic and Applications (FPL). IEEE, pp. 232–238.

Jain, Meenal and Gagandeep Kaur (2021). “Distributed anomaly detection using concept drift detection based hybrid ensemble techniques in streamed network data”. In: Cluster Computing 24, pp. 2099–2114.

Jain, Meenal, Gagandeep Kaur, and Vikas Saxena (2022). “A K-Means clustering and SVM based hybrid concept drift detection technique for network anomaly detection”. In: Expert Systems with Applications 193, p. 116510.

Kaspersky (2023). Rising threats: cybercriminals unleash 411,000 malicious files daily in 2023. Accessed: 2024-05-21. url: https://www.kaspersky.com/about/press-releases/2023%5C_rising-threats-cybercriminals-unleash-411000-malicious-files-daily-in-2023.

Leon, Miguel, Tijana Markovic, and Sasikumar Punnekkat (2022). "Comparative evaluation of machine learning algorithms for network intrusion detection and attack classification". In: 2022 international joint conference on neural networks (IJCNN). IEEE, pp. 01–08.

Lu, Jie et al. (2018). "Learning under concept drift: A review". In: IEEE transactions on knowledge and data engineering 31.12, pp. 2346–2363.

Maciel, Lucas Andrade, Matheus Alcântara Souza, and Henrique Cota de Freitas (2019). "Reconfigurable FPGA-based K-means/K-modes architecture for network intrusion detection". In: IEEE Transactions on Circuits and Systems II: Express Briefs 67.8, pp. 1459–1463.

Mahdi, Osama A et al. (2023). "Enhancing IoT Intrusion Detection System Performance with the Diversity Measure as a Novel Drift Detection Method". In: 2023 9th International Conference on Information Technology Trends (ITT). IEEE, pp. 50–54.

Martindale, Nathan, Muhammad Ismail, and Douglas A Talbert (2020). "Ensemble-based online machine learning algorithms for network intrusion detection systems using streaming data". In: Information 11.6, p. 315.

Murovic, Tadej and Andrej Trost (2019). "Massively parallel combinational binary neural networks for edge processing". In: Elektrotehniski Vestnik 86.1/2, pp. 47–53.

Murovič, Tadej and Andrej Trost (2020). "Resource-optimized combinational binary neural network circuits". In: Microelectronics Journal 97, p. 104724.

— (2021). "Genetically optimized massively parallel binary neural networks for intrusion detection systems". In: Computer Communications 179, pp. 1–10.

Ngo, Duc-Minh, Dominic Lightbody, et al. (2022). "HH-NIDS: heterogeneous hardware-based network intrusion detection framework for IoT security". In: Future Internet 15.1, p. 9.

Ngo, Duc-Minh, Andriy Temko, et al. (2021). "FPGA hardware acceleration framework for anomaly-based intrusion detection system in IoT". In: 2021 31st In-

ternational Conference on Field-Programmable Logic and Applications (FPL). IEEE, pp. 69–75.

Ngo, Duc-Minh, Binh Tran-Thanh, et al. (2019). "High-throughput machine learning approaches for network attacks detection on FPGA". In: Context-Aware Systems and Applications, and Nature of Computation and Communication: 8th EAI International Conference, ICCASA 2019, and 5th EAI International Conference, ICTCC 2019, My Tho City, Vietnam, November 28-29, 2019, Proceedings. Springer, pp. 47–60.

Nsunza, Watipatsa W, A-Q Ransford Tetteh, and Xiaojun Hei (2018). "Accelerating a secure programmable edge network system for smart classroom". In: 2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation. IEEE, pp. 1384–1389.

Pandit, Shraddha, Suchita Gupta, et al. (2011). "A comparative study on distance measuring approaches for clustering". In: International journal of research in computer science 2.1, pp. 29–31.

Pham-Quoc, Cuong, Tran Hoang Quoc Bao, and Tran Ngoc Thinh (2023). "FPGA/AI-powered architecture for anomaly network intrusion detection systems". In: Electronics 12.3, p. 668.

Qahtan, Abdulhakim A et al. (2015). "A pca-based change detection framework for multidimensional data streams: Change detection in multidimensional data streams". In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 935–944.

Ridder, Frank, Kuan-Hsun Chen, and Nikolaos Alachiotis (2023). "Accelerated Real-Time Classification of Evolving Data Streams using Adaptive Random Forests". In: 2023 International Conference on Field Programmable Technology (ICFPT). IEEE, pp. 232–237.

Roorda, Esther and Steven JE Wilton (2023). “Online Training from Streaming Data with Concept Drift on FPGAs”. In: 2023 24th International Symposium on Quality Electronic Design (ISQED). IEEE, pp. 1–8.

Roshan, Setareh et al. (2018). “Adaptive and online network intrusion detection system using clustering and extreme learning machines”. In: Journal of the Franklin Institute 355.4, pp. 1752–1779.

Seth, S, G Singh, and K Chahal (2021). “Drift-based approach for evolving data stream classification in Intrusion detection system”. In: Proceedings of the Workshop on Computer Networks & Communications, Goa, India, pp. 23–30.

Statista (2024). Statista - The Statistics Portal. url: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/.

Tavallaee, Mahbod et al. (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.

Todorov, Zdravko, Danijela Efnusheva, and T Nikolić (2021). “Fpga implementation of computer network security protection with machine learning”. In: 2021 IEEE 32nd International Conference on Microelectronics (MIEL). IEEE, pp. 263–266.

Togbe, Maurras Ulbricht et al. (2021). “Anomalies detection using isolation in concept-drifting data streams”. In: Computers 10.1, p. 13.

Ullah, Imtiaz and Qusay Mahmoud (May 2020). “A Scheme for Generating a Dataset for Anomalous Activity Detection in IoT Networks”. In: pp. 508–520. doi: 10.1007/978-3-030-47358-7_52.

Vreča, Jure et al. (2021). “Detecting network intrusion using binarized neural networks”. In: 2021 IEEE 7th World Forum on Internet of Things (WF-IoT). IEEE, pp. 622–627.

Wahab, Omar Abdel (2022). “Intrusion detection in the iot under data and concept drifts: Online deep learning approach”. In: IEEE Internet of Things Journal 9.20, pp. 19706–19716.

Xiaolan, Wang et al. (2022). “Evolving anomaly detection for network streaming data”. In: Information Sciences 608, pp. 757–777.

Yahyaoui, Aymen et al. (2021). “Machine learning based network intrusion detection for data streaming IoT applications”. In: 2021 21st ACIS International Winter Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD-Winter). IEEE, pp. 51–56.

Yang, Li and Abdallah Shami (2021). “A lightweight concept drift detection and adaptation framework for IoT data streams”. In: IEEE Internet of Things Magazine 4.2, pp. 96–101.

Zeng, Qingyu and Yuko Hara-Azumi (2024). “Hardware/Software Codesign of Real-Time Intrusion Detection System for Internet of Things Devices”. In: IEEE Internet of Things Journal.

Zou, Beiji et al. (2023). “Anomaly detection for streaming data based on grid-clustering and Gaussian distribution”. In: Information Sciences 638, p. 118989.