Drift-Mitigating Self-Learning System for Robust Encrypted Network Traffic Classification

Snehal Mishra (22BIT0325) and Priyanshi Saraf (22BIT0649)
Guide: Dr. Aswani Kumar Cherukuri
Project Type: Course Project
Course: Network and Information Security
Course ID: BITE401L
School of Computer Science Engineering and Information Systems
Vellore Institute of Technology
Deemed University, Vellore 632014, India

Abstract

Concept drift is a phenomenon where the statistical properties of target variables change over time and have a considerable implication on the performance of machine learning models deployed in network security environments. In the context of encrypted traffic analysis (ENTA), concept drift is evident in the changing relationships between the input features and target classifications, which leads to the performance of the model deteriorating if it is not appropriately addressed. Classifying anomalous encrypted network traffic is a critical challenge, especially in dynamic environments where concept drift changes in the statistical relationship between input features and target labels can significantly degrade model performance. The project tackles these issues by implementing a drift-mitigating self-learning hybrid system. It uses batch and online learning to detect and adapt to changing patterns in encrypted traffic. The dataset used for this is the CICIDS2017 dataset. The study also addresses the importance of selecting relevant features for accurate classification and effective drift detection. Concept drift is observed using statistical methods such as Kolmogorov–Smirnov test, while the real-time adaptation is done using the online learning model implemented with the River framework. The system consists of a batch-trained XGBoost classifier with an online Hoeffding Tree, allowing for continuous learning without full retraining. Comparative evaluations demonstrate that the batch-trained model has an accuracy of 92.32% without drift. However, under significant concept drift, the batch model has an accuracy of 87.36% while the online model has 82.84% accuracy and the hybrid model accuracy is 90.99%. After constant self-learning, the online model reaches 98.72% accuracy, with the hybrid model achieving 99.68% accuracy. This outcome indicates that the procedure is effective at preserving a high level of accuracy in identification among all traffic patterns, making it suitable for real-world deployment in dynamic network environments.

1. Introduction

In the contemporary digital age, the pervasiveness of encrypted communications has radically shifted the cybersecurity paradigm. Encryption mechanisms such as Transport Layer Security (TLS) and Hypertext Transfer Protocol Secure (HTTPS) are now the standard methodology for secure online transactions and communications. While encryption forms the core of confidentiality assurance and user privacy safeguarding, it also presents gigantic challenges to network security monitoring systems. Traditional deep packet inspection (DPI) methods, grounded in payload content examination, are rendered obsolete by encryption. Hence, Encrypted Traffic Analysis (ENTA) has emerged as a significant method to offer security in encrypted environments by analyzing flow-level metadata like packet lengths, timing features, and session durations without violating privacy constraints [10].

Although its potential, the efficacy of ENTA is severely undermined by the widespread application of static machine learning models under the assumption of a stationary data distribution. In practice, network environments are dynamic and constantly changing due to shifting user patterns, newly installed software, and emerging attack vectors. This is a phenomenon known as concept drift, causing inconsistencies between training data distribution and real-time traffic, ultimately causing static models' performance to decline over time [6], [7]. Different types of drift, such as gradual, abrupt, and periodic, have been witnessed in network traffic. For example, an evolving rise in encrypted traffic from a high-demanding application or a sudden rush in malicious traffic from a cyberattack can represent a change in the distribution of data that basic models are not able to catch up with [12], [8].

The impact of unabated concept drift is especially important for high-risk domains such as anomaly and intrusion detection systems (IDS). There, misclassification, false positives, or false negatives, can be critical in implication. In industrial control networks, healthcare facilities, and financial systems, failure to detect new or emerging threats because models are not up-to-date can lead to data leaks, service outages, and public safety hazards [11], [7]. Therefore, the stabilization of ENTA systems against non-stationary streams of data is not merely an issue of performance, but it's a matter of necessity to ensure system reliability and trust.

To counteract these limitations, research is increasingly in favor of adaptive learning architectures that are capable of real-time detection of and reaction to concept drift. Online learning, drift detection methods, and self-evolving models have been found to have potential in maintaining model accuracy and robustness in dynamic settings [5],[13],[14]. It remains an open problem to integrate these features into ENTA systems because of the high velocity, volume, and variability of encrypted network traffic.

In the current research, we introduce a new hybrid ENTA model that combines batch and online learning paradigms. The model utilizes drift detection mechanisms and incremental model updates to ensure high classification accuracy despite changing traffic patterns. By crossing the divide between static analysis and dynamic adaptation, our solution seeks to provide a resilient and scalable solution for real-time encrypted traffic anomaly detection for contemporary cybersecurity infrastructures.

Problem Statement and Objectives:

The core research issue is the difficulty of sustaining efficient and precise anomaly detection in encrypted network traffic under concept drift conditions. Conventional machine learning algorithms for ENTA are trained on static data, with the expectation that traffic feature classification relationships do not change over time. Real-world network environments are dynamic with evolving attack patterns, user behaviors, and application protocols, leading to concept drift. This drift degrades the performance of static models, leading to increased false negatives (attacks that are missed) and false positives (false alarms), hence undermining the reliability of network security systems.

The issue is compounded by the requirement of maintaining data confidentiality via encryption, which restricts the application of deep packet inspection methods. Thus, the models have no choice but to depend on metadata and traffic flow patterns, which, in turn, keep changing as network behavior progresses. This issue needs to be met with an adaptive system that can identify the concept drift, find relevant features, and make real-time updates to the models in order to keep accuracy high.

To solve the research problem, the following specific objectives have been formulated:

1. Analyze and Select Best Features: Compare different traffic features to choose the most suitable ones for proper classification and efficient drift detection. The choice should be optimally driven by features having discriminative power in anomaly detection and responsiveness towards the impact of concept drift. Automatic feature selection techniques will be investigated to be able to respond to changing traffic patterns, trying to achieve a trade-off between model simplicity and forecasting accuracy.

2. Compute and Track Class Distributions: Compute the distribution of normal and anomalous traffic classes across the CICIDS2017 dataset and track how distributions evolve over time. Distribution changes between classes can signal the onset of concept drift and information regarding the kinds of adaptation required to keep the model accurate. Statistical procedures will be used to compute class probability shifts and to determine when retraining or adaptation processes should be invoked.

3. Compare Accuracy Under Drift Scenarios: Perform a comparative assessment of model performance under three different scenarios. Under static conditions (no drift), a baseline level of accuracy is determined using a dataset with no induced concept drift. This is used as a reference point to measure the effect of drift in later scenarios. Under concept drift without adaptation, the performance degradation suffered by the model is measured to assess the susceptibility of static methods to changing traffic patterns. Lastly, after adaptation with the self-learning system, the efficacy of the introduced hybrid solution in reducing concept drift impact is tested. This includes comparing the accuracy of the adapted model with the baseline accuracy and the performance of the non-adaptive model under drift, highlighting the capability of the system to restore or even sustain high accuracy.

4. Integrate Self-Learning and Retraining Mechanisms: Create a hybrid system that integrates a batch-trained XGBoost classifier with an online Hoeffding Tree model to facilitate continuous learning and adaptation. The system will have mechanisms for concept drift detection and initiating retraining or adaptation of the models. This goal is to develop a system that can automatically adapt to evolving traffic patterns without human intervention.

By meeting these goals, this study will present a solution to achieving high accuracy in anomaly detection in dynamic encrypted network environments, thus enhancing the security and reliability of network infrastructure.

2. Background and Related Work

In conventional machine learning frameworks, the relationship between output labels and input features is often assumed to be stationary over time. This, however, does not hold true in most real-world environments, especially in Encrypted Network Traffic Analysis (ENTA), where dynamic and non-stationary conditions are the default. The loss of model performance as a result of alterations in this statistical relationship is known as concept drift [7]. In the case of ENTA, such drift is caused by changing user behavior, adversary strategies, and changes in application-layer protocols. These include adjustments in attack vectors by attackers, rolling out new encryption mechanisms, and the creation of new services that distort traffic patterns [6],[15].

2.1 ENTA Systems Concept Drift

Concept drift appears in a number of ways: sudden, where the change happens all at once (e.g., the introduction of a new type of attack); gradual, where old and new patterns coexist during the transition; incremental, with gradual but persistent changes; and recurring, where past concepts cyclically reappear. ENTA systems are especially susceptible to such forms due to their dependency on metadata and behavior-based features such as flow size, inter-arrival time, and session length, which are heavily influenced by user behavior and network-level changes [8]. Static models are inherently incapable of keeping up with such evolution, leading to a rise in false negatives (stealthy attacks) and false positives (normal flows signaled as threats). This not only attacks the credibility of intrusion detection systems but also provokes alarm fatigue and security blind spots. In high-assurance environments such as financial institutes, healthcare, or government networks, these misclassifications are likely to generate major disruptions and vulnerabilities. Thus, learning models that are adaptive and able to learn from the changing data stream in real-time are very

important for maintaining detection accuracy and system robustness [8].

2.2 Challenges Posed by Encryption

Future networks increasingly use encryption protocols to provide confidentiality of data as well as compliance with regulations. Protocols like TLS 1.3 and DNS-over-HTTPS (DoH) have made classical Deep Packet Inspection (DPI) mostly ineffective through packet payload reduction of visibility. Consequently, anomaly detection systems now have to resort to coarse-grained traffic features such as packet timing, packet size distributions, and flow-level metadata [16],[17]. Yet these characteristics are themselves context-sensitive and subject to variation in response to changing user behavior, application usage, or network settings, making consistent classification much harder. Added to this is the fact that encryption technologies are themselves dynamic. They change with the deployment of new handshake protocols, encrypted SNI (Server Name Indication), and padding strategies, each of which changes perceivable traffic attributes. This two-pronged challenge providing correct detection in the absence of payload information yet being resilient to changing traffic patterns, highlights the paramount importance of adaptive techniques in encrypted traffic analysis.

2.3 Drift detection and adaptation techniques

To decrease the effect of concept drift, researchers have applied a range of detection and adaptation techniques. Detection algorithms for drift typically occur in two forms:

- **Explicit Drift Detectors**: They use statistical tests to compare distributions over time. One popular one is ADWIN (Adaptive Windowing), which dynamically adapts a sliding window and warns drift if statistically significant performance variation is observed. Some other methods include the Kolmogorov-Smirnov test, which calculates the greatest distance between empirical cumulative distributions [8].
- **Implicit Drift Detectors**: These infer drift from model performance decline, e.g., increasing error rate or decreasing confidence in predictions. These are useful in situations where there is sparsely labeled data but inference streaming is ongoing.

Online learning algorithms are the obvious choice for adaptation. The Hoeffding Tree (Domingos & Hulten, 2000) and its variants (e.g., Adaptive Random Forests) can adapt internal models incrementally as new instances are seen. These models are well-suited to scenarios where there are constraints in terms of resources as they are time and memory-efficient. The ensemble learning strategy of having multiple models trained on different time slices, but discarding low-performing ones and keeping better ones is another promising methodology. Robustness to drift and diversity are enabled by the nature of such architecture.

Through the incorporation of detection and adaptation mechanisms within a single framework, real-time responsiveness is enabled, which is critical for ENTA systems. Modern implementations in libraries like River and scikit-multiflow enable integration, with drift-aware classifiers that can be applied in streaming contexts.

2.4 Self-Learning Systems for ENTA

Self-improving systems are a milestone for ENTA, as they introduce autonomy in the cycle of updating the model. Instead of periodic retraining, these systems keep learning with data, whether labeled or not, that keeps pouring in using strategies like incremental learning, semi-supervised learning, and confidence-based pseudo-labeling. One of the first in this area was the Incremental SVM (ISVM) proposed by Laskov [18], which is updated with new examples coming online. Although computationally expensive, ISVM offered an initial proof of concept for security-related continual learning. More current architectures take advantage of stream mining frameworks, including those provided by River, for lightweight and module-based updates. These have been used on online implementations of Naïve Bayes, k-Nearest Neighbours, and linear models with success depending on the feature engineering and type of drift involved [19].

Deep learning techniques have also been used for encrypted traffic classification. Lotfollahi [17] introduced Deep Packet, a framework based on convolutional networks with time-based flow features. Though it learned well from benchmark datasets with high accuracy, its static nature exposed it to performance loss in the presence of drift. Anderson [16] investigated the effect of adversarial drift and established that even very well-trained models deteriorate rapidly unless they learn online. Their results further strengthened the need for ongoing learning mechanisms to ensure robustness.

Although promising, self-learning systems suffer from a principal problem: a lack of available labeled data within real-time environments. This has generated interest in semi-supervised learning methods whereby the model makes pseudo-label assignments to unlabelled data based on high-confidence predictions. Methods including confidence-based sampling, self-training loops, and co-training work to enhance label efficiency while preventing error propagation.

2.5 Summary of Literature

The literature on concept drift in network traffic analysis shows that static models fail to cope with shifts in network patterns and suffer from declining performance. Concept drift affects both accuracy of detection and boosts false positives, which diminishes the reliability of intrusion detection systems. A variety of drift detection algorithms, such as ADWIN and the Kolmogorov-Smirnov test, are explained, and adaptation techniques encompass online learning algorithms such as Adaptive Random Forests and Hoeffding Trees. Self-learning systems such as those using stream mining libraries like River provide real-time model update but their application of big annotated datasets is problematic. Furthermore, advances in ensemble-based and semi-supervised learning methods have the potential to provide more flexible and scalable solutions. Overall, the research highlights the need for scalable, flexible, and adaptive models to effectively cope with drift in dynamic network environments.

3. Dataset Description

The CICIDS2017 dataset is a large network traffic dataset developed by the Canadian Institute for Cybersecurity, intended for testing network intrusion detection systems (IDS). The dataset includes both normal and malicious traffic samples, gathered under various network attack conditions. It is widely utilized to test the performance of anomaly detection systems, such as encrypted network traffic analysis (ENTA) systems.

Table 1: Characteristics of the CICIDS2017 dataset

Dataset Name	CICIDS2017
Dataset Type	Multi-class
Year of release	2017
Total number of instances	2,830,540
Number of features	83
Number of distinct classes	15

3.1 Dataset Overview

- Traffic Types: The dataset captures a variety of network traffic environments, including normal traffic and a variety of attacks like DoS, DDoS, Botnet, and Port Scanning.
- Number of Instances: The dataset comprises about 80 million network traffic records in different attack and normal traffic environments.
- Number of Features: The dataset features 80+ features, with traffic-related descriptors like packet length, flow duration, and packet timing.

Table 2: Distribution of Traffic Classes in CICIDS2017 Dataset

Traffic Type	**Number of instances**	**Percentage (%)**
BENIGN	2359087	80.3%
DoS Hulk	231072	8.2%
PortScan	158930	5.6%
DDoS	41835	4.5%
DoS GoldenEye	10293	0.4%
FTP-Patator	7938	0.3%
SSH-Patator	5897	0.2%
DoS slowloris	5796	0.2%

DoS Slowhttptest	5499	0.2%
Bot	1966	<0.1%
Web Attack – Brute Force	1507	<0.1%
Web Attack – XSS	652	<0.1%
Infiltration	36	<0.01%
Web Attack – Sql Injection	21	<0.01%
Heartbleed	11	<0.01%
Total	**2830743**	**100%**

3.2 Feature Engineering and Selection

The CICIDS2017 dataset offers more than 80 features derived from packet-level and flow-level network traffic. Yet, most of the raw features are redundant, irrelevant to classification, or may bring noise and overfitting to the model, particularly under the condition of class imbalance and data drift. Thus, meticulous feature engineering and selection were conducted to improve the performance and stability of the model.

Features that have high mutual information with the class label were kept, and features that have near-zero variance or a high correlation (>0.95) were discarded. The selected top 10 features include statistics concerning flow duration, packet sizes, inter-arrival times, and header flags. These features were robustly effective in discriminating between attack and benign traffic.

Table 3: Top Selected Features and their Relevance

Feature Name	**Rationale**
Flow Duration	Indicates session time; short for DoS/DDoS
Total Fwd Packets	Useful for identifying scanning and brute-force
Bwd Packet Length Mean	Captures reply pattern; varies for exfiltration
Flow IAT Std	Measures inter-packet delay variation
Fwd Packet Length Std	High deviation in burst attacks
Down/Up Ratio	Detects unbalanced traffic like flooding
Flow Bytes/s	Indicator of bulk transfer (e.g., DDoS)
Flow Packets/s	Helps spot port scanning and flooding
Bwd IAT Mean	Timing-based anomaly detection
Fwd IAT Total	Session start burst indication

All of these features were further processed employing batch learning and online standard scaling (through River.preprocessing.StandardScaler) for the streaming model. These features were not only statistically relevant but also interpretable, making them effective for both supervised and streaming models with drift detection.

4. System Architecture

The intended Network Intrusion Detection System (NIDS) architecture supports both batch learning and online learning to detect intrusions from network traffic. The architecture consists of three essential layers: the Preprocessing Drift Detection Layer, the Self-Learning & Retraining Layer, and the Classification Layer. The system adapts dynamically to new traffic patterns and addresses potential data drift in real-time.

4.1 Preprocessing Drift Detection Layer

This layer takes care of data cleaning, feature scaling, and transformation prior to passing the data to the models. It also checks when the distribution of the incoming data changes, utilizing techniques such as ADWIN (Adaptive Windowing) and KS-test (Kolmogorov-Smirnov test). These methods track the streams of data for any drastic shifts in traffic patterns, which would indicate drift.

4.2 Self-Learning & Retraining Layer

Here the system learns in a cumulative fashion from new instances. In the event of drift, the model can be updated or relearned using incremental learning or a batch of fresh data. The Hoeffding Tree algorithm is employed in real-time online learning, whereas batch models may

be relearned from time to time. Windowing methods are utilized in the system for efficient handling of data and maintaining the model as current as recent traffic patterns.

4.3 Classification Layer

different categories (e.g., benign, DoS, DDoS, botnet traffic). The batch model is utilized for big-data analysis, and the online model is utilized to provide real-time classification and make adjustments to new data dynamically.

As shown in Figure 1, the data flows through the preprocessing layer where it is cleaned and scaled, followed by drift detection. Once the drift is identified, the self-learning layer updates the model incrementally. Finally, the classification layer outputs predictions for each network traffic sample.

Two models within this layer are utilized for traffic classification: XGBoost for batch learning and Hoeffding Tree for online learning. The two models work to classify the network traffic into

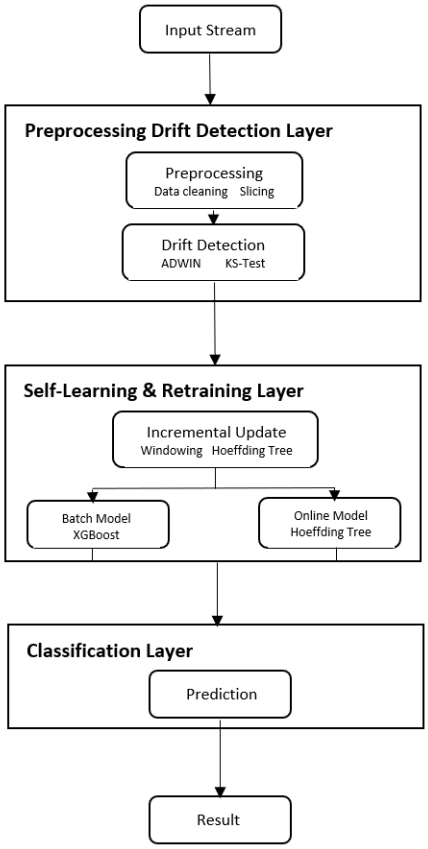

Figure 1. System Architecture Diagram

5. Methodology

The system integrates batch and online learning methods to construct a strong Network Intrusion Detection System (NIDS) that can detect attacks in real-time and learn to accommodate changing network trends.

5.1 Drift Detection Techniques

To ensure long-term model usability in fluctuating network environments, the system maintains two complementary drift detection techniques:

Kolmogorov–Smirnov (KS) Test: The KS test is a statistical method used to check whether the distribution of selected attributes in the incoming data stream is not equal to a specified reference set. If the test's p-value is less than a specified threshold (usually 0.05), the feature is indicated as having drifted. This allows for early detection of distributional shifts, usually reflective of changing and varying attack patterns.

ADWIN (Adaptive Windowing): For streaming classification online detection of concept drift, ADWIN tracks the prediction error by the model for a sliding window. If an increase in error rate above some statistically significant ratio is observed, a concept drift is asserted. ADWIN maintains dynamic adjustment to the window's size based on the stability in the arriving instances, hence suited for high-speed environments.

5.2 Classification Models

Two classifiers are implemented within a hybrid framework that combines high initial accuracy with responsiveness, a Batch Classifier and an Online Classifier.

Batch Classifier – XGBoost: It is a gradient-boosting decision tree model that is learned from past network traffic. SelectFromModel and StandardScaler are used before inference for feature selection and scaling. XGBoost provides a solid baseline on tabular data and is also well-suited for class imbalance.

Online Classifier – Hoeffding Tree (River): For self-learning and streaming inference, a Hoeffding Tree is employed. The decision boundaries of this model are updated incrementally with each new sample, without retraining over the entire dataset. As more data are available, the tree structure adapts, which makes it well-suited for continuous, real-time classification.

5.3 Self-Learning Strategy

The framework facilitates online learning through Incremental Learning, Windowing Techniques, and Online Standardization.

Incremental Learning: When processing every new labeled sample, the online classifier updates itself through the use of the learn_one() method and adjusts its internal structure and weights without requiring full model retraining.

Windowing Technique: In conjunction with ADWIN, the windowing mechanism constantly tries to minimize prediction errors. When drift is detected, the model can focus on up-to-date patterns by funneling learning in the new window, which in effect allows rapid adaptation to new threats.

Online Standardization: To provide consistency in input distributions to the online model, real-time standardization is utilized through River. preprocessing.StandardScaler. This enables the system to normalize and process features on the fly, thus making them compatible with the expectations of the model even in concept drift.

This hybrid methodology ensures that the system benefits from the strong predictive power of offline-trained models while maintaining adaptability and responsiveness through online learning and drift management. By coupling batch and streaming methodologies together, the system is able to provide reliable performance under static as well as dynamic network conditions.

6. Experimental Analysis

This section outlines the experimental setup, tools utilized, evaluation metrics, and detailed analysis of the model performance under various drift scenarios.

6.1 Tools and Libraries

The experimental setup was deployed with a mixture of batch and stream learning libraries. Batch training was done through Scikit-learn for evaluation and preprocessing, and XGBoost for gradient-boosting classification. For online learning, River was utilized to support stream-based updates through the Hoeffding Tree classifier and ADWIN drift detector. Pandas and NumPy were utilized for data manipulation, and SciPy was used to perform the Kolmogorov–Smirnov (KS) test for statistical drift detection. Streamlit was the real-time interface for interaction, visualization, and testing of the model.

6.2 Experimental Setup

The batch model was trained beforehand with an 80-20 train-test split on the CICIDS2017 dataset. Feature scaling was done with StandardScaler, and top features were chosen using variance thresholding and mutual information ranking. A reference sample was kept aside for detecting drift, against which the incoming samples were compared with the KS-test ($\alpha = 0.05$). For online learning, the incoming data stream was processed sample by sample. The model was updated with actual labels (if present) and predictions were tracked in real time. The drift was monitored

continuously through ADWIN and automatic notifications for concept drift.

6.3 Evaluation Metrics

In order to properly evaluate model performance, a number of standard classification metrics were employed. Accuracy is a ratio of correct predictions to total predictions, a metric of overall model accuracy. Precision is a ratio of true positive predictions to total predicted positives, a metric of the precision of the model. Recall (Sensitivity) is a measure of the ratio of true positives predicted to actual positives and is a measure of how well the model can identify relevant instances. F1-Score is the harmonic mean of Precision and Recall that balances the trade-off between them. Area Under the ROC Curve (AUC) measures the trade-off between true positive rate and false positive rate for different thresholds and evaluates the discriminative power of the model. The Confusion Matrix gives a detailed breakdown of prediction results, facilitating error pattern analysis.

These metrics were computed separately for the batch model, the online learning model, and the hybrid model in order to compare performance under different drift scenarios. Graph and table presentations also illustrate model performance across drift and retraining cycles.

6.4 Accuracy Under Drift Conditions

To analyze the proposed Network Intrusion Detection System (NIDS) robustness under changing network conditions, model performance was measured for three different scenarios: No Drift, Significant Drift, and Retraining. Accuracy was calculated for the batch-trained model (XGBoost), the online learning model (Hoeffding Tree with ADWIN drift detector), and the hybrid model that chooses predictions of the more confident classifier.

Table 4: Accuracy Under Drift Conditions

Scenario	Batch Accuracy (%)	Online Accuracy (%)	Hybrid Accuracy (%)
No Drift	92.32	75.52	92.39
Significant Drift	87.36	82.84	90.99
With Retraining	87.36	98.72	99.68

As observed, although the batch model has consistent performance in drift-free scenarios, its accuracy declines under drift. Conversely, the online model shows better adaptability, especially upon retraining. The hybrid model consistently shows the highest accuracy in all scenarios, indicative of its capacity to combine the strengths of both learning paradigms.

6.5 Classification Metrics

A detailed breakdown of classification metrics across conditions further illustrates the effectiveness of the online learning framework. The *No Drift* case reveals severe class imbalance issues, where the model only successfully identifies benign traffic and fails to detect any attack classes. However, with the introduction of significant drift and subsequent retraining, the model's ability to generalize across all classes improves remarkably.

Table 5: Classification metrics (Without Drift)

Class	Precision (%)	Recall (%)	F1-Score (%)	Support
BENIGN	75.52	100.00	86.05	15103
Bot	0.00	0.00	0.00	13
DDoS	0.00	0.00	0.00	1099
DoS GoldenEye	0.00	0.00	0.00	93
DoS Hulk	0.00	0.00	0.00	2126
DoS Slowhttptest	0.00	0.00	0.00	50
DoS slowloris	0.00	0.00	0.00	50
PortScan	0.00	0.00	0.00	1465

Macro Average: Precision = 9.44%, Recall = 12.50%, F1-Score = 10.76%
Weighted Average: Precision = 57.03%, Recall = 75.52%, F1-Score = 64.99%

Table 6: Classification metrics(with Significant Drift)

Class	Precision (%)	Recall (%)	F1-Score (%)	Support
BENIGN	83.22	97.13	89.63	15104
Bot	0.00	0.00	0.00	13
DDoS	98.67	33.85	50.41	1099
DoS GoldenEye	55.32	27.96	37.14	93
DoS Hulk	73.27	45.91	56.45	2126
DoS Slowhttptest	23.53	16.00	19.05	50
DoS slowloris	0.00	0.00	0.00	50
PortScan	98.29	35.22	51.86	1465

Macro Average: Precision = 54.04%, Recall = 32.01%, F1-Score = 38.07%
Weighted Average: Precision = 83.57%, Recall = 82.84%, F1-Score = 80.48%

Table 7: Classification metrics (After Retraining)

Class	Precision (%)	Recall (%)	F1-Score (%)	Support
BENIGN	99.43	99.19	99.31	15104
Bot	44.44	30.77	36.36	13
DDoS	99.09	99.55	99.32	1099
DoS GoldenEye	79.41	58.06	67.08	93
DoS Hulk	94.40	99.81	97.03	2126
DoS Slowhttptest	100.00	28.00	43.75	50
DoS slowloris	100.00	36.00	52.94	50
PortScan	99.05	99.45	99.25	1465

Macro Average: Precision = 89.48%, Recall = 68.85%, F1-Score = 74.38%
Weighted Average: Precision = 98.72%, Recall = 98.72%, F1-Score = 98.62%

6.6 Confusion Matrix Analysis

The confusion matrices per scenario reassert the quantitative outcomes above. During No Drift, the model shows high bias in predicting a single benign class. During Significant Drift, detection is improved for prevalent types of attacks like DDoS and DoS Hulk but low-frequency attacks like Bot and slowloris still remain under-detected. After retraining, the confusion matrix shows successful discrimination across almost all categories of attack, demonstrating online learning model adaptation.

6.7 Graphical Comparison

A line plot was created to compare and visualize model accuracy for all drift scenarios. The graph shows that the batch model works consistently in stable settings but is not adaptable during drift. Conversely, the online model improves over time with increased exposure and retraining, eventually reaching 99.02% accuracy after adaptation. The hybrid approach, which integrates the strengths of batch and online learning, outperforms the individual methods consistently, with an accuracy of 99.69% after retraining.

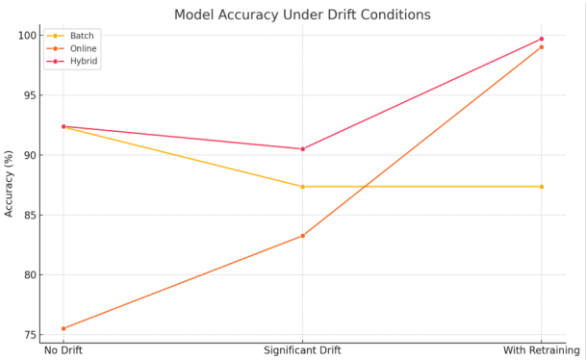

Figure 1. Line plot comparing Model Accuracy under drift conditions

6.8 Interpretation and Insights

The experiment validates the requirement for drift-sensitive learning in dynamic settings. As much as the batch model performs well in environments that do not change, it trails behind owing to drift. Online learning mitigates this through constant adaptation, albeit temporarily at a lower rate of accuracy. The hybrid solution addresses these limitations, yielding better performance under both drift and non-drift.

6.9 Effectiveness of Feature Selection and Drift Management

The feature selection approach enhanced training efficiency and model generalization. KS-test was found to be useful in identifying minor and substantial changes in the underlying distribution. ADWIN accurately detected concept drift, allowing timely adjustments for the online learner. Combining these modules facilitated strong detection and prevention of drift, making it possible to deploy the intrusion detection system on a large scale and with robustness.

Conclusions

This project targets the primary challenge of encrypted network traffic analysis in concept drift scenarios, which largely impedes the functioning of static machine learning algorithms for real-world evolving networks. The main contribution is the development and deployment of a hybrid self-improving Network Intrusion Detection System (NIDS) incorporating batch learning using an XGBoost classifier, combined with online learning through the Hoeffding Tree

algorithm employed in the River framework. Such a design supports the system's ability to maintain high classification precision while continuously updating to changing traffic patterns without a need for retraining.

One of the most important contributions of this work is the use of strong and complementary drift detection methods—Kolmogorov–Smirnov (KS) test and Adaptive Windowing (ADWIN)—which allow for the early detection of distributional shifts in traffic attributes. This enables timely model adaptation and ensures long-term stability of performance. In addition, the system puts special emphasis on feature selection and real-time standardization, which are critical in maintaining the model input stable and enhancing drift detection sensitivity. Empirical experiments based on the CICIDS2017 dataset demonstrate the effectiveness of the presented method, and the hybrid model achieves up to 99.68% accuracy following infinite self-learning, dominating stand-alone static or online models significantly in concept drift conditions. All these results enhance the system's scalability, robustness, resilience, and practicality toward real-time analysis of encrypted traffic.

Briefly, this research project presents a novel, adaptive, and scalable hybrid solution to intrusion detection in encrypted networks. By effective handling of concept drift and the combination of the strengths of both batch and streaming approaches, the proposed system offers a strong and future-proof solution to securing network infrastructures against constantly evolving cyberattacks.

Code and Datasets

The source code used for the experiments and model implementation is available at the following GitHub Repository Link: https://github.com/SnehalMishra02/concept-drift-mitigation-in-ENTA.

Google Colab Link: https://colab.research.google.com/drive/1-hEUEyjAk4-fodT8qSvNES6-93BpMrYl?usp=sharing

The dataset used in this study is CICIDS2017, which is publicly available at: https://www.unb.ca/cic/datasets/ids-2017.html.

References

[1] T. Bakhshi and B. Ghita, “Anomaly detection in encrypted internet traffic using hybrid deep learning,” *Security and Communication Networks*, vol. 2021, Article ID 5363750, 2021. [Online]. Available: https://doi.org/10.1155/2021/5363750

[2] Z. Chen, G. Cheng, J. Li, T. Qin, Y. Zhou, and X. Luan, “Drift-oriented self-evolving encrypted traffic application classification,” *arXiv preprint*, arXiv:2501.04246, 2025. [Online]. Available: https://arxiv.org/abs/2501.04246

[3] J. Li, K. Malialis, and M. M. Polycarpou, “Autoencoder-based anomaly detection with incremental learning,” *arXiv preprint*, arXiv:2305.08977, 2023. [Online]. Available: https://arxiv.org/abs/2305.08977

[4] M. Soltani, K. Khajavi, M. J. Siavoshani, and A. H. Jahangir, “A multi-agent adaptive deep learning framework for online intrusion detection,” *Cybersecurity*, vol. 6, no. 1, Article 10, 2023. [Online]. Available: https://doi.org/10.1186/s42400-023-00199-0

[5] J. Zhu, S. Cai, F. Deng, B. C. Ooi, and W. Zhang, “METER: A dynamic concept adaptation framework for online anomaly detection,” *Proc. VLDB Endow.*, vol. 17, no. 5, pp. 794–807, 2023. [Online]. Available: https://doi.org/10.14778/3636218.3636233

[6] I. Žliobaite, “Learning under concept drift: An overview,” *arXiv preprint*, arXiv:1010.4784, 2010. [Online]. Available: https://arxiv.org/abs/1010.4784

[7] G. Widmer and M. Kubat, "Learning in the Presence of Concept Drift and Hidden Contexts," *Machine Learning*, vol. 23, no. 1, pp. 69–101, 1996. [Online]. Available: https://doi.org/10.1023/A:1018046501280

[8] J. Gama, I. Žliobaite, M. Pechenizkiy, and M. M. Gaber, “A survey on concept drift adaptation,” *ACM Comput. Surv.*, vol. 46, no. 4, pp. 1–37, 2014. Available: https://doi.org/10.1145/2523813

[9] T. R. Hoens, R. Polikar, and N. V. Chawla, “Learning from streaming data with concept drift,” *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 10, pp. 1724–1737, 2012. Available: https://link.springer.com/article/10.1007/s13748-011-0008-0

[10] A. H. Lashkari, G. Draper-Gil, M. S. I. Mamun, and A. A. Ghorbani, “Characterization of tor traffic using time-based features,” in *Proc. Int. Conf. Inf. Syst. Secur. Priv.*, Cham: Springer, 2017, pp. 430–441. Available: https://www.scitepress.org/Papers/2017/61056/

[11] G. I. Webb, R. Hyde, H. Cao, S. Agarwal, and J. Vreeken, “Characterizing concept drift,” *Data Min. Knowl. Discov.*, vol. 30, no. 4, pp. 964–994, 2016. Available: https://doi.org/10.1007/s10618-015-0448-4

[12] H. Lu *et al.*, “Learning under Concept Drift: A Review,” *IEEE Trans. Knowl. Data Eng.*, 2018. Available: https://doi.org/10.1109/TKDE.2018.2876857

[13] P. Nagendhiran and L. Kuppusamy, “Adaptive drift detection method (ADDM) for data streams,” *J. King Saud Univ. Comput. Inf. Sci.*, vol. 32, no. 6, pp. 655–664, 2020. Available: https://ideas.repec.org/a/wsi/jikmxx/v20y2021i01ns0219649221500088.html

[14] A. Pesaranghader, H. L. Viktor, and E. Paquet, “McDiarmid Drift Detection Methods for evolving data streams,” *Pattern Recognit.*, vol. 76, pp. 449–461, 2018. Available: https://arxiv.org/abs/1710.02030

[15] A. Tsymbal, “The Problem of Concept Drift: Definitions and Related Work,” Department of Computer Science, Trinity College Dublin, Tech. Rep. TCD-CS-2004-15, 2004. [Online]. Available: https://www.scss.tcd.ie/publications/tech-reports/reports.04/TCD-CS-2004-15.pdf

[16] Anderson, B., Paul, S., & McGrew, D. (2016). *Deciphering Malware's Use of TLS (Without Decryption)*. arXiv preprint arXiv:1607.01639. Available: https://arxiv.org/abs/1607.01639

[17] Lotfollahi, M., Siavoshani, M. J., Shirali Hossein Zade, R., & Saberian, M. (2020). *Deep Packet: A Novel Approach for Encrypted Traffic Classification Using Deep Learning*. Soft Computing, 24, 1999–2012. Available: https://doi.org/10.1007/s00500-019-04030-2

[18] Laskov, P., Gehl, C., & Müller, K. R. (2006). Incremental support vector learning: Analysis, implementation, and applications. *Journal of Machine Learning Research*, 7, 1909–1936. Available: https://www.jmlr.org/papers/volume7/laskov06a/laskov06a.pdf

[19] Montiel, J., Halford, M., Mastelini, S. M., Bolmier, G., Sourty, R., Vaysse, R., Zouitine, A., Gomes, H. M., Read, J., Abdessalem, T., & others. (2021). River: Machine learning for streaming data in Python. *Journal of Machine Learning Research*, 22, Article 1380. Available: https://www.jmlr.org/papers/volume22/20-1380/20-1380.pdf