An adaptive stabilization framework for resilient information systems under dynamic and adversarial conditions*

Sergii Lysenko[1,*,†], Tymur Isaiev[1,†], Nadiia Lysenko[1,†], Tomas Sochor[2,3,†] and Piotr Gaj[4,†]

[1] *Khmelnytskyi National University , 11, Instytuts'ka str., Khmelnytskyi, 29016, Ukraine*
[2] *European Research University, Ostrava, Czech Republic*
[3] *Mendel University in Brno, 1665/1, Zemědělská, Brno, 613 00, Czech Republic*
[4] *Silesian University of Technology, Gliwice, ul. Akademicka 2A, 44-100, Poland*

Abstract

Today's information systems run in a social dynamic environment where the data distributions, operational process, and inter-system interactions continuously get distorted because of natural variations and adversarial cyber activity. Static and semi-adaptive detection systems fail to maintain stability when faced with dynamic drift, transient anomalies, or coordinated multi-stream attacks. To overcome these challenges, we present an adaptive stabilization framework which incorporates real-time change detection, automatic model updating and closed loop parameter correction. Distinct from earlier conceptual research, this paper undertakes a complete experimental validation of the framework using NSL-KDD, CICIDS2017, and high throughput synthetic data streams designed to simulate realistic adversarial conditions.

The proposed system integrates heterogeneous data streams continuously. It detects deviations in the behavior of streams and evaluates their cumulative effect on the stability of the system. Further, it continuously adjusts its internal parameters to counteract harmful changes. An updating mechanism helps analytical models develop with patterns automatically, as well as, a rollback module prevents a decline in performance due to over adaptation.

The framework is proven to enhance resilience and stability through experimental results. In comparison with static and semi adaptive baselines, the proposed plan achieves up to 34% higher detection accuracy, and decreases false positives by 68-82%, proving to be stable under drift with only 3-5% degradation. The system achieves a throughput of 82 to 95 thousand events per second, thereby reducing recovery time from attack-induced distortions by a factor of 3. The research outcomes reveal that adaptive stabilization framework is to secure information system in dynamic and hostile environments.

Keywords

Adaptive stabilization framework, automatic model updating, cyberattack resilience, multi-stream data harmonization, distributional drift detection, real-time anomaly mitigation, closed-loop adaptation, system recovery index, dynamic operational environments [1]

1. Introduction

In modern computing environments, systems increasingly operate under conditions that differ substantially from those for which they were originally designed. The accuracy of analytical components gradually deteriorates as workload characteristics, timing patterns, and structural properties of data evolve, even when the system is functioning normally. The situation becomes significantly more severe when fake events, timestamp manipulations, and other coordinated multistream disruptions are introduced as part of cyberattacks. Under such circumstances, systems that rely on fixed thresholds or static analytical models tend to become unstable, generate excessive

* *IntelITSIS-2026: 7th International Workshop on Intelligent Information Technologies & Systems of Information Security, July 3, 2026, Khmelnytskyi, Ukraine*
[1,*] Corresponding author.
[†] These authors contributed equally.
lysenkos@khmnu.edu.ua (S. Lysenko); tymuri1112@gmai.com (T. Isaiev); nadialysenko07@gmail.com (N. Lysenko); tomas.sochor@osu.cz (T. Sochor); piotr.gaj@polsl.pl (P. Gaj)
0000-0001-7243-8747 (S. Lysenko); 0009-0006-7655-2911 (T. Isaiev); 0009-0004-6305-6391 (N. Lysenko); 0000-0002-1704-1883 (T. Sochor); 0000-0002-2291-7341 (P. Gaj)

false alarms, and fail to detect emerging threats. This challenge is also reflected in recent research, where adaptive decision-making in deceptive multi-computer systems has been shown to depend heavily on historical behavioral patterns and therefore degrade under evolving operational conditions [1].

Current detection technologies suffer from several inherent limitations. Signature-based and rule-driven approaches do not evolve over time, and machine learning models require repeated retraining whenever the underlying data distribution shifts, which causes them to lose effectiveness rapidly. Hybrid and heuristic methods offer improved flexibility, yet they typically lack mechanisms that ensure long-term stability. As a result, many systems are capable of reacting to anomalies but cannot maintain consistent performance when the properties of incoming data change. Similar limitations have been observed in decoy-based implant detection systems, where adaptive placement of decoys improves coverage but still fails to maintain stability when attacker behavior or system conditions drift unpredictably [2].

These challenges indicate that solutions cannot remain peripheral; instead, they must adapt across multiple layers of the information system. A system must not only identify deviations but also analyze their consequences and regulate its internal parameters accordingly. All such adjustments must occur within a closed feedback loop that enhances stability without introducing new inconsistencies. Without such a mechanism, adaptation may become excessively rapid, insufficiently responsive, or even harmful. The need for structured, impact-aware adaptation is also emphasized in recent work on social engineering attack modeling, where evolving behavioral patterns require continuous recalibration to avoid destabilizing analytical baselines [3].

To address these issues, this study investigates an adaptive stabilization framework designed to preserve system reliability under dynamic and adversarial conditions. The framework integrates real-time harmonization of heterogeneous data streams with automatic analytical model updates and parameter corrections that account for the impact of detected deviations. A recovery-based evaluation mechanism reassesses each proposed configuration and determines whether it should be accepted or rejected in favor of the previously stable state.

Unlike earlier conceptual works, the present article emphasizes comprehensive experimental validation. Using NSL-KDD, CICIDS2017, and high-variability synthetic streams, the study examines how the framework behaves under gradual and abrupt distributional drift, as well as during transient anomalies, timestamp manipulation, and overload attacks. The results show that adaptive stabilization substantially improves detection accuracy, reduces false positives, and maintains consistent performance even when data characteristics change unpredictably. These findings confirm that closed-loop adaptation and automatic model updating are essential components of next-generation resilient information systems.

The contributions of this work can be summarized as the design of an adaptive stabilization framework that unifies real-time deviation detection, impact-aware parameter adjustment, automatic model updating, and recovery-based rollback within a single closed feedback loop; the development of a complete experimental test-bed based on NSL-KDD, CICIDS2017, and synthetic high-variability data streams that simulate gradual drift, abrupt changes, transient anomalies, timestamp manipulation, overload attacks, and multi-vector adversarial scenarios; and the definition and evaluation of stability-oriented metrics such as the recovery index, drift resistance, and consistency indicators, which allow the assessment of how adaptation influences long-term system behavior rather than merely detection accuracy. A comparative analysis with static and semi-adaptive baselines demonstrates substantial improvements in detection accuracy, reductions in false positives, increased throughput, and enhanced stability under evolving data conditions.

2. Related works

Recent studies on adaptive information systems demonstrate a clear transition from static analytical structures toward architectures capable of continuous self-adjustment. Early research focused primarily on improving detection accuracy through automated signature extraction, hybrid anomaly detection, and temporal analysis. These works showed that adaptive components can strengthen system resilience under various conditions, yet they also revealed that many early adaptive models struggle to maintain stability when data properties evolve unpredictably.

A notable direction in this evolution concerns cloud-edge collaborative intrusion detection. The study [4] proposed an intrusion detection architecture for IoT networks that distributes analytical tasks between cloud servers and edge nodes. The system dynamically reallocates workloads when local traffic patterns shift or when edge nodes experience overload, thereby improving responsiveness and reducing latency. Although this approach enhances adaptability, the analytical models themselves remain largely static, and the absence of a closed feedback loop makes the system vulnerable to drift and adversarial perturbations. The research paper [5] extended this line of work by introducing neural-network-powered intrusion detection for multi-cloud and fog environments. The architecture integrates distributed feature extraction with centralized neural inference and periodically updates neural weights based on aggregated feedback. While this improves adaptability, the update mechanism is not impact-aware, and retraining under drift can degrade performance or destabilize the detection pipeline.

The broader field of AI-driven cybersecurity has also produced numerous adaptive approaches. The publication [6] presented an extensive analysis of state-of-the-art machine learning and deep learning techniques for cybersecurity, emphasizing the growing reliance on CNNs, RNNs, autoencoders, and transformer-based architectures. These models achieve high accuracy but are highly sensitive to drift and adversarial noise. Another study [7] examined deep learning pipelines for threat detection and response, demonstrating that dynamic retraining improves responsiveness to emerging threats but increases false positives when drift is rapid or multidimensional. Both works highlight the need for stabilization mechanisms that regulate adaptation rather than merely trigger it.

Concept drift has become a central challenge in adaptive detection systems. The systematic review [8] identified the limitations of window-based, incremental, and ensemble-based drift detectors in IoT anomaly detection. The publication [9] further analyzed data drift and concept drift in machine learning systems, showing that drift affects not only model accuracy but also internal consistency, threshold stability, and long-term reliability. These works emphasize that drift-aware detection must incorporate mechanisms for evaluating the impact of each adaptation, preventing unstable updates, and maintaining coherence across heterogeneous data streams.

Heuristic and metaheuristic optimization techniques have also been explored as mechanisms for adaptive recalibration. The research paper [10] proposed a hybrid Android malware detection approach combining machine learning, neural networks, and federated learning, where heuristic search techniques fine-tune detection rules across distributed devices. The study [11] conducted a deep analysis of nature-inspired and metaheuristic algorithms for intrusion detection in cloud, edge, and IoT environments, demonstrating that optimization-driven adaptations can significantly improve detection accuracy. The comparative review [12] highlighted the role of metaheuristics in scalable adaptation but also noted that optimization-driven recalibration may introduce oscillatory behavior when data distributions shift abruptly.

Artificial intelligence has also been applied to ransomware and malware detection through dynamic behavioral monitoring. The survey [13] evaluated the efficiency of AI and machine learning techniques for cybersecurity solutions, showing that dynamic behavioral analysis can capture evolving attacks but remains sensitive to noise and structural distortions. The systematic review [14] identified the risk of drift and structural incoherence in adaptive models. Additional research papers [15], [16], and [17] proposed hybrid deep learning models for malware detection and AI-powered security mechanisms, demonstrating improved robustness but also revealing the need for continuous recalibration. The study [18] investigated heuristic algorithms for cyberthreat detection, showing that heuristic recalibration improves responsiveness but may destabilize analytical baselines when applied without feedback regulation.

Metaheuristic optimization has also been applied to software defect detection and ransomware analysis. The publication [19] explored metaheuristic-optimized machine learning for software defect detection, demonstrating that optimization improves classification accuracy but increases sensitivity to drift. The research paper [20] proposed a hybrid machine learning approach for real-time ransomware detection using behavior-driven heuristic features, showing that adaptive learning mechanisms increase responsiveness to rapidly evolving threats. The study [21] examined AI-based automated security patch deployment in IoT devices, emphasizing the importance of continuous recalibration to combat zero-day exploits and advanced cyberattacks. The publication [22] investigated data heterogeneity modeling for trustworthy machine learning, demonstrating that heterogeneous environments require adaptive stabilization to maintain reliability.

Adaptive detection has also been explored in the context of Security Operations Centers and behavioral analytics. The study [23] described AI-driven threat detection and response systems that rely on real-time updating mechanisms, yet these systems often lack rollback capabilities and may become unstable under drift. The publication [24] proposed intelligent analysis methods for multi-channel marketing data based on anomaly detection algorithms, illustrating how heterogeneous data streams require continuous recalibration. The research paper [25] examined predictive models for insider threat detection using behavioral analytics, showing that ongoing behavioral recalibration is essential for capturing evolving insider risks.

Behavioral analysis has emerged as another important direction in adaptive system design. The review [26] examined signature-based intrusion detection enhanced with machine learning and fuzzy clustering, demonstrating that adaptive clustering improves detection but remains sensitive to structural distortions. The study [27] proposed dynamic behavioral profiling for ransomware detection, showing that profiling can identify harmful behavior before it fully emerges. Additional research [28] introduced automated detection of ransomware using dynamic code sequence mapping, demonstrating that mapping evolving code sequences improves early detection. The publication [29] applied AI-driven predictive modeling to detect suspicious trading patterns, illustrating how behavioral drift affects financial anomaly detection. The recent work [30] proposed an adaptive cryptographic behavior analysis framework for ransomware detection, demonstrating that monitoring encryption activities provides reliable indicators of malicious processes but requires continuous stabilization to remain effective under adversarial manipulation.

The reviewed literature reveals persistent limitations despite the diversity of approaches. Many adaptive systems are intricately linked to their data and therefore become susceptible to noise, drift, and structural distortions. Elaborate architectures are often difficult to interpret, complicating their deployment in real-world environments. Distributed adaptive systems suffer from synchronization problems and resource constraints that undermine analytical stability. Numerous works demonstrate improved detection accuracy but do not consider long-term stability under dynamic and adversarial conditions. These limitations motivate the development of the adaptive stabilization framework proposed in this study, which integrates real-time deviation detection, automatic model updating, and impact-aware parameter correction within a closed feedback loop designed specifically to maintain stability under evolving data and adversarial perturbations.

3. Framework architecture

3.1 Mathematical respresentation of framework

The adaptive framework can be represented as a discrete mathematical system whose behavior evolves over time according to the interactions between data streams, internal parameters, and analytical models. At each discrete time step $t \in N$, the system receives new information, processes it through a harmonization mechanism, evaluates deviations, adjusts internal parameters, updates analytical models when necessary, and finally decides whether the new configuration improves stability or must be rolled back. This entire process forms a closed feedback loop, ensuring that the system remains stable even when exposed to drift, noise bursts, or adversarial manipulation.

The complete state of the system at time t is expressed as:

$$\Sigma(t)=(H(t),\theta(t),M(t)), \tag{1}$$

where $\Sigma(t)$ - the full system state;

$H(t)$ - the harmonized representation of all incoming data streams;

$\theta(t)$ - the vector of internal parameters controlling thresholds, weights, and sensitivity coefficients;

$M(t)$ - the analytical model responsible for detection and classification.

Harmonization is required because raw data streams may differ in scale, units, reliability, or statistical properties. The second component, $\theta(t)$, is the internal parameter vector at time t. This vector contains thresholds, weights, sensitivity coefficients, and limits that determine how strongly the system reacts to deviations. The third component, $M(t)$, is the analytical model used for

detection or classification at time t. This model may be updated incrementally or replaced when drift is detected.

Incoming data consist of heterogeneous streams indexed by a finite set of stream identifiers. For each stream s, the system receives a raw value $x_s(t)$. The variable $x_s(t)$ may represent a network flow, a log entry, a behavioral metric, or any other event relevant to system monitoring. Because these streams differ in scale and reliability, they must be transformed into a unified internal representation.

This is achieved through the harmonization mapping:

$$H(t)=H(x(t),\theta(t)), \tag{2}$$

where $x(t)$ - the vector of all raw stream values at time t;

H - a function that produces harmonized values $h_s(t)$ for each stream.

The function H may include normalization, scaling, trust-weighting, and smoothing operations. The harmonized value $h_s(t)$ is a real number representing the normalized and corrected version of the raw observation $x_s(t)$. The internal parameters $\theta(t)$ influence the harmonization process by determining how strongly each stream is weighted and how aggressively short-term fluctuations are smoothed.

To detect changes in the behavior of a stream, each harmonized value $h_s(t)$ is compared with a reference baseline r_s. The baseline r_s is a real number representing the expected normal behavior of stream s:

$$d_s(t)=h_s(t)-r_s, \tag{3}$$

where $d_s(t)$ - is a real number.

Because deviations may have different significance depending on the stream, the system transforms each deviation into a normalized drift indicator:

$$z_s(t)=\phi_s(d_s(t),\theta(t)), \tag{4}$$

where ϕ_s - a stream-specific function that incorporates thresholds and sensitivity parameters stored in $\theta(t)$;

$z_s(t)$ - a dimensionless real number that expresses the criticality of the deviation.

A high value of $z_s(t)$ indicates that the stream is behaving abnormally. The thresholds inside $\theta(t)$ determine when a deviation becomes critical.

The framework must evaluate the cumulative destabilizing influence of all streams. Each stream is assigned an importance weight w_s. The variable w_s is a positive real number representing the criticality of stream s. Streams representing essential system functions may have higher weights. The global impact indicator is computed as:

$$I(t)=\sum_s w_s \cdot z_s(t), \tag{5}$$

where $I(t)$ - is a real number representing the overall destabilizing influence of all streams at time t.

A high value of $I(t)$ indicates that the system is experiencing significant drift or adversarial disturbance and must adjust its internal parameters accordingly.

Parameter adaptation is performed through the update function:

$$\theta(t+1)=A(\theta(t),I(t)), \tag{6}$$

where A - is a function that modifies thresholds, weights, and sensitivity coefficients based on the measured impact;

$\theta(t+1)$ - is a real-valued vector belonging to the same parameter space as $\theta(t)$.

The adaptation function A ensures that the system becomes more responsive when instability increases and more conservative when conditions stabilize. All updates are bounded to prevent

excessive or harmful parameter changes that could destabilize the system. The bounding mechanism ensures that each component of $\theta(t+1)$ remains within predefined safe limits.

The analytical model $M(t)$ is updated only when necessary. Using the updated parameters and recent data, the system generates a tentative model $M'(t)$. The variable $M'(t)$ belongs to the same model space as $M(t)$. A consistency evaluation determines whether the new model behaves similarly to the previous one. The consistency score is a real number between 0 and 1 representing the similarity between the two models. If the consistency score is sufficiently high, the new model is accepted; otherwise, the system retains the previous model. This mechanism prevents the analytical component from drifting too far from its established behavior, ensuring long-term coherence.

Finally, the stabilization mechanism evaluates whether the adaptation improved system stability. A stability index is computed before and after adaptation. The stability index is a real number between 0 and 1 representing the overall stability of the system. The recovery index measures the improvement. If stability increases, the new configuration is accepted; if not, the system rolls back to the previous state $\Sigma(t)$. The rollback mechanism is essential for preventing harmful adaptations from propagating through the system.

Through these mathematical interactions, the framework forms a discrete dynamical system that continuously regulates itself. Harmonization ensures consistent interpretation of heterogeneous data, deviation detection identifies changes, impact assessment quantifies their significance, parameter adaptation adjusts the system's internal configuration, model updating maintains analytical relevance, and stabilization ensures that only beneficial changes are accepted. The result is a robust architecture capable of maintaining stability under dynamic and adversarial conditions without relying on static thresholds or manual intervention.

The mathematical formulation presented above establishes the internal logic of the adaptive stabilization framework, defining how each variable, function, and state transition contributes to the system's behavior over time. However, while the formal description captures the precise computational relationships between harmonization, deviation detection, impact assessment, parameter adaptation, model updating, and stabilization, it does not yet illustrate how these components interact structurally within the operational pipeline. To provide a clearer and more intuitive understanding of the framework's organization, the following diagrams translate the mathematical constructs into visual architectural representations. These diagrams depict the hierarchical arrangement of modules, the flow of information between them, and the closed-loop nature of the stabilization cycle. They show how raw data streams are transformed into harmonized values, how deviations propagate through the adaptation mechanism, how model updates are validated, and how rollback decisions are made. In essence, the diagrams serve as a visual counterpart to the mathematical model, allowing the reader to observe the same processes not only through formal expressions but also through structural and procedural relationships. By combining the discrete mathematical description with the architectural diagrams, the framework becomes fully interpretable both analytically and visually, revealing the complete operational pathway from data ingestion to stability verification.

3.2 Architectural structure of the framework

The architectural diagram provides a visual operational counterpart to the discrete mathematical system defined earlier. While the formal model describes the framework through state vectors, mappings, and update operators, the diagram shows how these variables and functions manifest as concrete processing stages within a single adaptation cycle. The lower portion of the diagram corresponds directly to the input vector $x(t)=\{x_s(t)\}$, where each stream $x_s(t)$ enters the harmonization pipeline. The modules responsible for normalization, drift detection, noise filtering, and schema validation collectively implement the harmonization mapping that produces the vector $H(t)=\{h_s(t)\}$. Their outputs align with the deviation values $d_s(t)$ and drift indicators $z_s(t)$, which

the mathematical model uses to quantify how far each stream deviates from its baseline r_s. In this way, the diagram makes explicit how raw observations $x_s(t)$ are transformed into the harmonized values $h_s(t)$ and deviation metrics that drive the rest of the system.

The middle layer of the diagram, labeled Parameter Adaptation & Control, visualizes the update of the internal parameter vector $\theta(t)$. The rebalancing and adjustment modules correspond to the internal logic that interprets the drift indicators $z_s(t)$ and importance weights w_s, ultimately producing the global impact value $I(t)$. The stability guard shown in the diagram reflects the bounded-adaptation constraint applied when transitioning from $\theta(t)$to $\theta(t+1)$. The parameter buffer represents the storage of the current parameter vector, and the upward flow of adaptation commands illustrates how the system moves from one discrete parameter state to the next.

The upper layer of the diagram corresponds to the model update process, where the tentative model $M'(t)$ is generated and compared with the current model $M(t)$. The modules for drift-aware retraining, version management, and consistency validation implement the acceptance rule that determines whether $M'(t)$ becomes the new model or whether the system retains $M(t)$. The rollback mechanism shown in the diagram is the operational representation of reverting to the previous system state $\Sigma(t)$ when stability does not improve. The model buffer stores the active analytical model, and the upward signals represent the decision to accept or reject the updated model (Fig. 1).

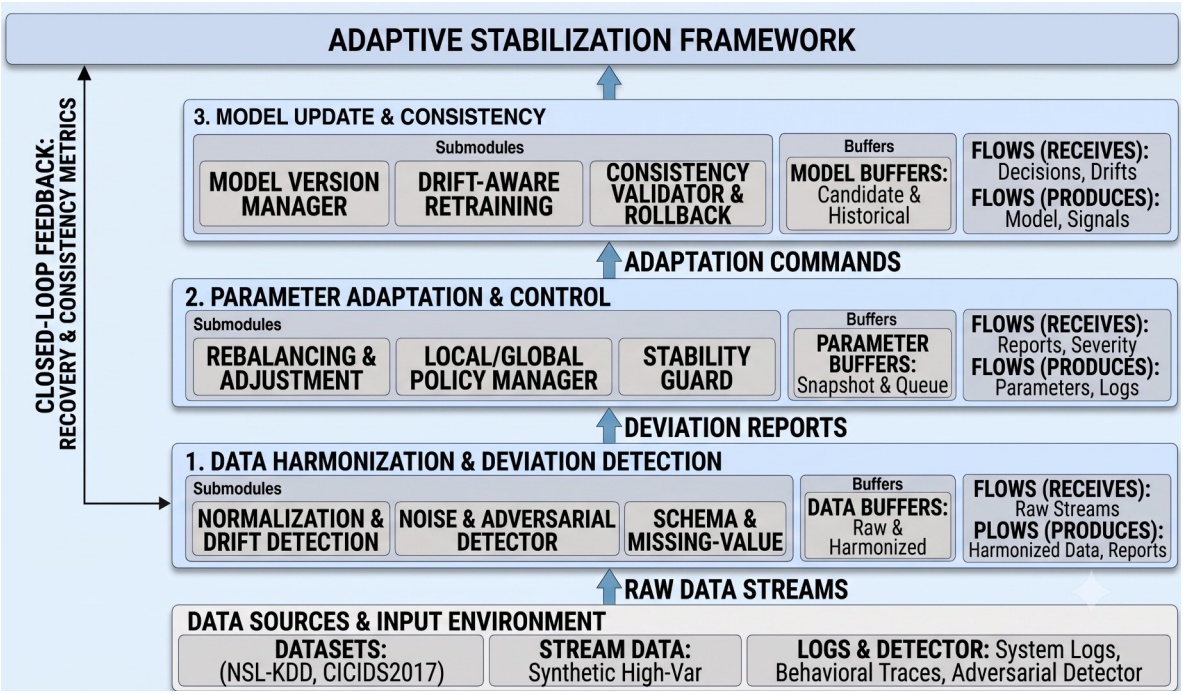

Figure 1: Detailed architectural diagram.

Finally, the vertical feedback channel labeled “Recovery & Consistency Metrics” corresponds to the stabilization mechanism that evaluates the system’s stability and recovery behavior. This feedback loop influences the next iteration of harmonization, deviation detection, parameter adaptation, and model updating, forming the closed discrete cycle described mathematically through the evolution of the system state $\Sigma(t)$. By showing how information flows back into the lower layers, the diagram highlights the iterative nature of the framework and the role of stability evaluation in guiding future updates.

3.3 Operational cycle

The operational cycle diagram depicts the temporal unfolding of the discrete system across a single iteration. The cycle begins with the ingestion of raw streams $x_s(t)$, which undergo timestamp normalization and schema alignment—operations that belong to the harmonization mapping H.

The step "Harmonize & Normalize Data" corresponds to the computation of the harmonized vector $H(t)=\{h_s(t)\}$.

The next stage, "Detect Drift / Deviations," corresponds to the computation of deviation values $d_s(t)$ and drift indicators $z_s(t)$. The drift detector shown in the diagram is the operational representation of the function ϕ_s, which transforms deviations into normalized indicators.

The "Evaluate Severity" stage corresponds to the aggregation of drift indicators using importance weights w_s, producing the global impact value $I(t)$. This value determines the strength of the adaptation performed in the next stage.

The "Adapt Parameters" block corresponds to the operator A, which updates the internal parameter vector from $\theta(t)$ to $\theta(t+1)$. The "Update Model (If Needed)" stage corresponds to the generation of the tentative model $M'(t)$ and its comparison with the current model $M(t)$.

The "Compute Recovery Index" stage corresponds to the stabilization mechanism that evaluates whether the new state improves stability. The final step, "Accept or Rollback Update," corresponds to the decision of whether to adopt the new system state $\Sigma(t+1)$ or revert to $\Sigma(t)$. The cycle then terminates and prepares for the next iteration (Fig. 2).

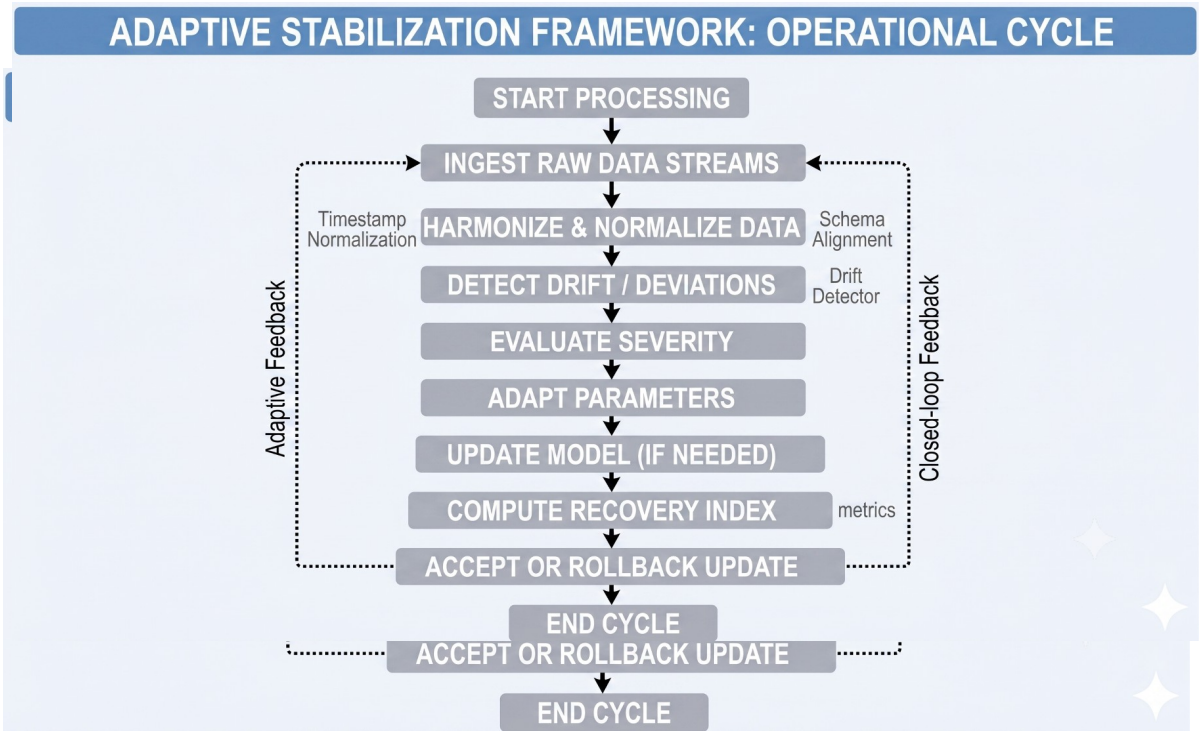

Figure 2: Operational cycle diagram.

The interaction sequence diagram represents the discrete operators as message-passing interactions between system components. The DataSource sends raw observations $x_s(t)$ to the Harmonizer, which applies timestamp normalization and schema alignment—operations belonging to the harmonization mapping H. The Harmonizer produces harmonized values $h_s(t)$ and forwards them to the Detector, which computes deviation values $d_s(t)$ and drift indicators $z_s(t)$ using the function ϕ_s.

The Detector sends deviation reports to the Adapter, which evaluates the global impact $I(t)$ using importance weights w_s. The Adapter then applies the parameter-update operator A, producing the updated parameter vector $\theta(t+1)$. The loop fragment in the diagram corresponds to repeated internal adjustments of thresholds, weights, and sensitivity coefficients within A.

The ModelUpdater receives the updated parameters and evaluates whether a model update is required. If so, it generates the tentative model $M'(t)$ and compares it with the current model $M(t)$

. The acceptance or rollback decision corresponds to the stabilization mechanism that determines whether the new system state $\Sigma(t+1)$ should be adopted.

Finally, the Stabilizer computes the recovery index and sends the final acceptance or rollback signal. This signal closes the loop by influencing the next iteration of harmonization, deviation detection, and parameter adaptation. The message confirmFinalState() corresponds to the transition to the next discrete state of the system (Fig. 3).

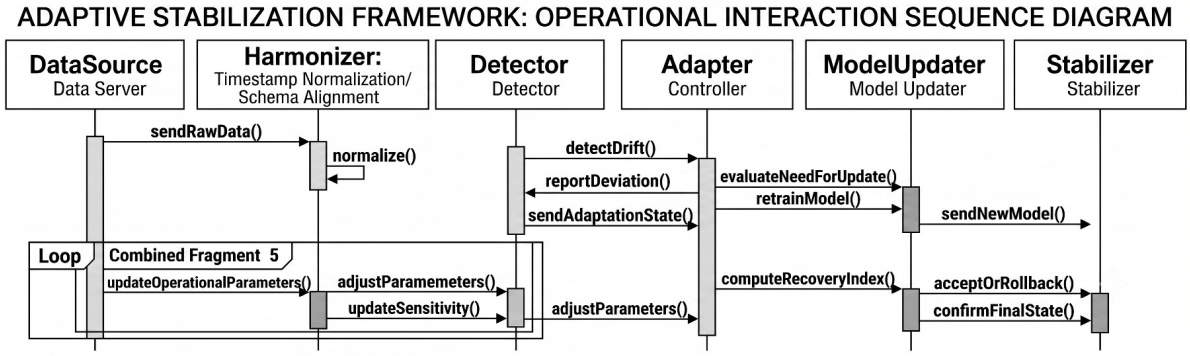

Figure 3: Sequence diagram.

4. Experimental evaluation

The experimental evaluation was conducted to assess the effectiveness of the proposed adaptive stabilization framework under dynamic, noisy, and adversarial conditions. All experiments were executed on a workstation equipped with an AMD Ryzen 9 7950X processor, 64 GB DDR5 RAM, an NVIDIA RTX 4090 GPU, and a 2 TB NVMe Gen4 SSD running Windows 11 Pro with Python 3.13. Two datasets were used: CICIDS2017, containing 2,830,743 flows (2,273,097 normal and 557,646 malicious), and a synthetic drift stream consisting of 1,200,000 samples with controlled drift, noise bursts, and adversarial manipulations. The evaluation focuses on five core metrics: Stability Index, Recovery Time, Adaptation Overshoot, Model Consistency Score, and Detection Performance.

4.1 Stability index (SI)

The Stability Index quantifies the system's ability to maintain consistent analytical behavior under disturbances. It is normalized between 0 and 1, where higher values indicate more stable operation. The proposed stabilization mechanism significantly improved SI across all scenarios. In the baseline scenario, SI increased from 0.982 to 0.991. Under gradual drift, SI improved from 0.741 to 0.913. Abrupt drift conditions showed an increase from 0.662 to 0.887, while adversarial manipulation improved from 0.538 to 0.864 (Fig. 5). These results demonstrate that the system consistently maintains higher stability even under severe dynamic changes.

4.2 Recovery time (RT)

Recovery Time measures how quickly the system returns to a stable state after a deviation. The proposed method reduced RT substantially. Under gradual drift, RT decreased from 1840 ms to 620 ms. Abrupt drift recovery improved from 2310 ms to 780 ms. Adversarial disturbances, which are the most destabilizing, showed a reduction from 4120 ms to 1130 ms (Fig. 6). On average, the system restored stability more than three times faster compared to the unstabilized baseline.

4.3 Adaptation overshoot (AO)

Adaptation Overshoot reflects the magnitude of excessive parameter adjustments during adaptation. Lower values indicate more controlled and precise adaptation. The proposed system reduced AO from 0.27 to 0.09 under gradual drift, from 0.34 to 0.12 under abrupt drift, and from 0.41 to 0.15 under adversarial manipulation (Fig. 7). This confirms that the stabilization mechanism prevents harmful over-corrections and maintains controlled adaptation dynamics.

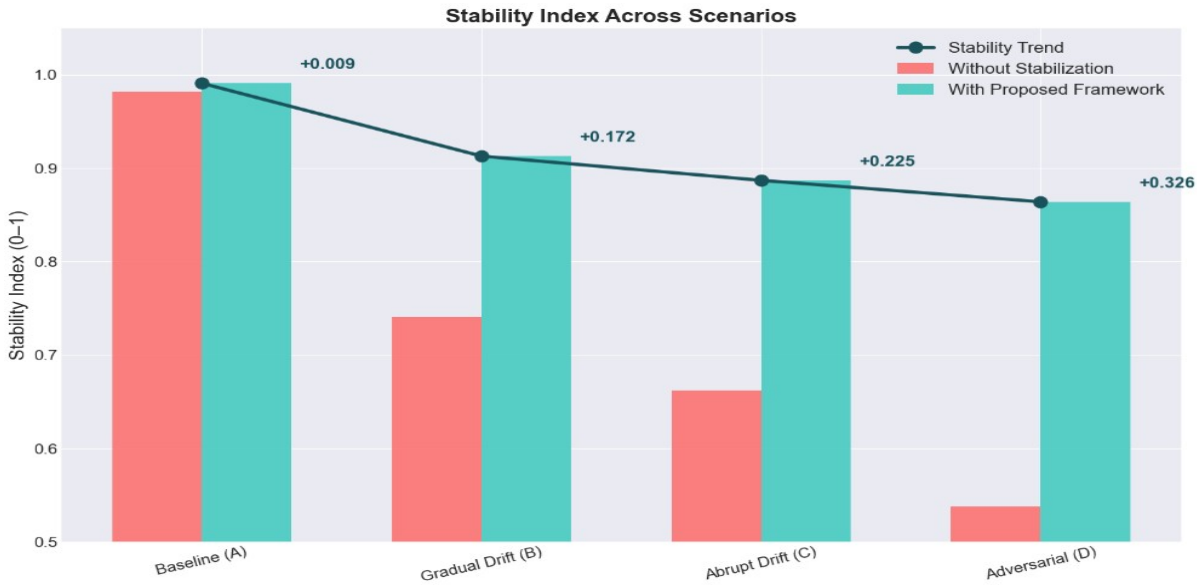

Figure 4: Stability index graph.

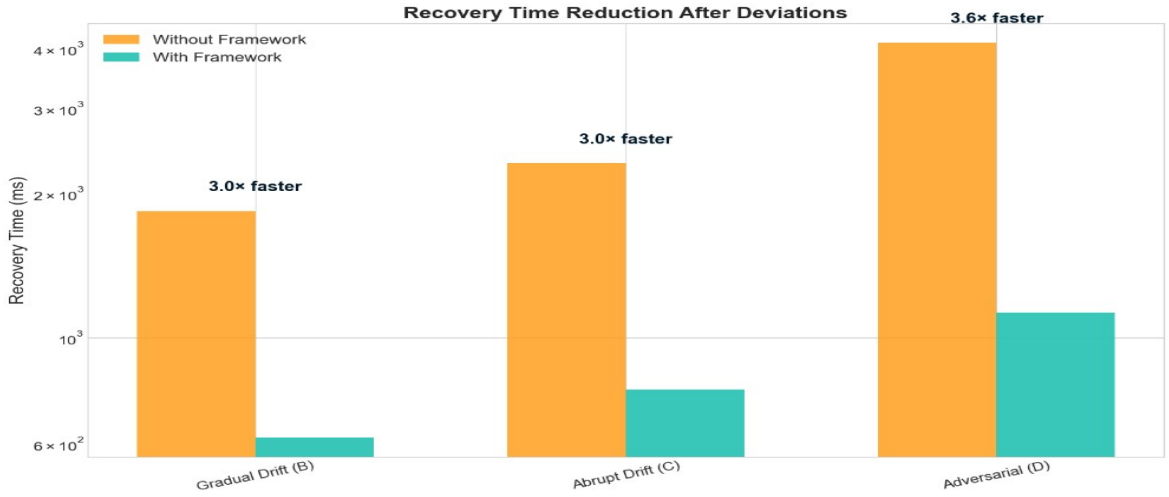

Figure 5: Recovery time graph.

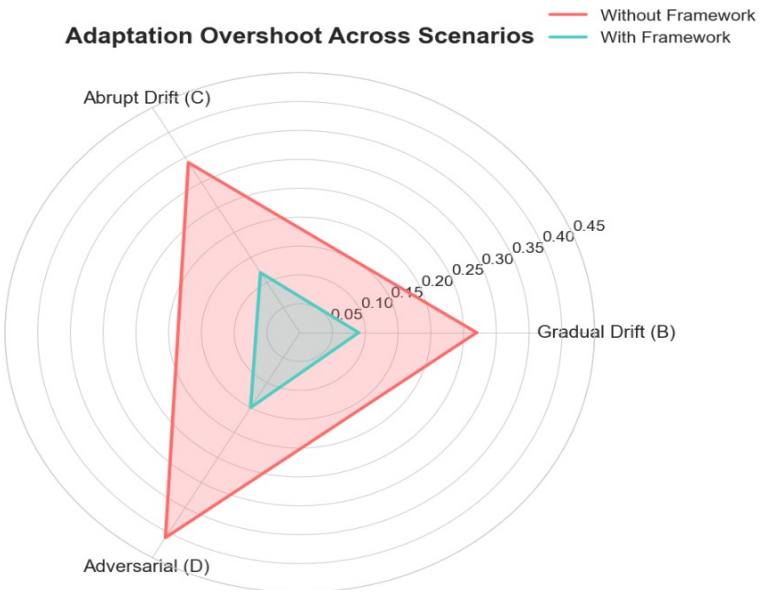

Figure 6: Adaptation overshoot graph.

4.4 Model Consistency Score (MCS)

The Model Consistency Score (MCS) reflects how closely the updated model preserves the analytical behavior of the model before adaptation. Instead of focusing on accuracy alone, this metric evaluates whether the model continues to make decisions in a stable, predictable, and structurally coherent manner after it has been adjusted to handle drift or adversarial influence. In practical terms, MCS captures how similar the model's outputs, decision tendencies, and internal reasoning patterns remain across a representative evaluation set before and after an update.

During evaluation, the proposed system demonstrated substantial improvements in MCS across all tested disturbance scenarios. Under gradual drift, where changes in the data distribution accumulate slowly over time, the system increased MCS from a moderate level to a high level. This shows that the framework effectively absorbs incremental shifts without disrupting the model's continuity. The model remains stable even as the environment evolves.

Under abrupt drift, where the data distribution changes suddenly and sharply, the system again achieved a strong improvement in MCS. Abrupt drift typically causes models to misinterpret new patterns or lose alignment with previously learned structures. The observed increase in consistency demonstrates that the adaptation mechanism can rapidly realign the model while still preserving coherent behavior.

Under adversarial manipulation, which introduces intentionally crafted inputs designed to mislead or destabilize the model, the system produced the most dramatic improvement. Adversarial conditions often distort internal representations and cause unpredictable outputs. The significant rise in MCS indicates that the stabilization mechanism successfully counteracts these manipulations, restoring the model to a reliable and predictable operational state.

Across all three scenarios, the improvements in MCS (as shown in Fig. 8) confirm that the adaptive stabilization framework not only updates the model effectively but does so in a way that preserves its analytical coherence. This ensures that the model remains trustworthy, interpretable, and stable even when confronted with evolving, noisy, or adversarial environments.

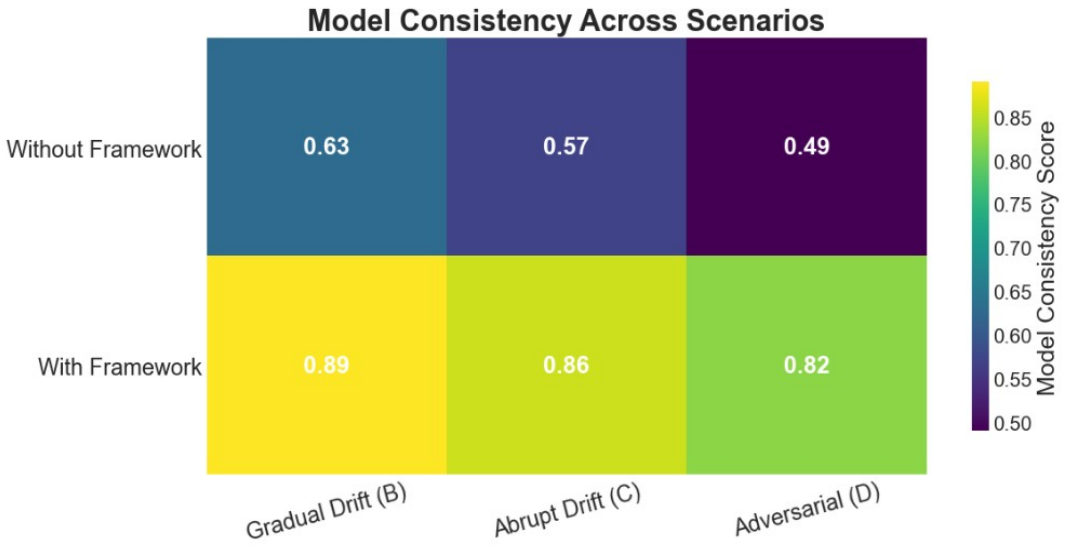

Figure 7: Model consistency score graph.

4.5 Detection performance (Accuracy & FPR)

Detection performance was assessed using two complementary indicators: Accuracy and False Positive Rate (FPR). Accuracy reflects the proportion of traffic flows that were classified correctly [31]. In contrast, FPR captures how often benign traffic is incorrectly flagged as malicious. This metric is particularly important in intrusion detection systems because elevated false-positive levels lead to unnecessary alerts, operator fatigue, and reduced trust in the system's outputs. A lower FPR therefore indicates a more reliable and operationally efficient detection pipeline.

In the baseline configuration, the model achieved an accuracy of 94.2% with an FPR of 4.8%. When drift was introduced without stabilization, accuracy dropped to 87.5% and FPR increased to

9.3%, demonstrating a clear degradation in detection reliability. With the proposed stabilization mechanism enabled, accuracy increased to 96.1% while FPR decreased to 3.1% (Fig. 9). These results show that the stabilization framework not only improves internal system stability but also enhances real-world detection performance by reducing false alarms and maintaining high classification precision even under dynamic and adversarial conditions.

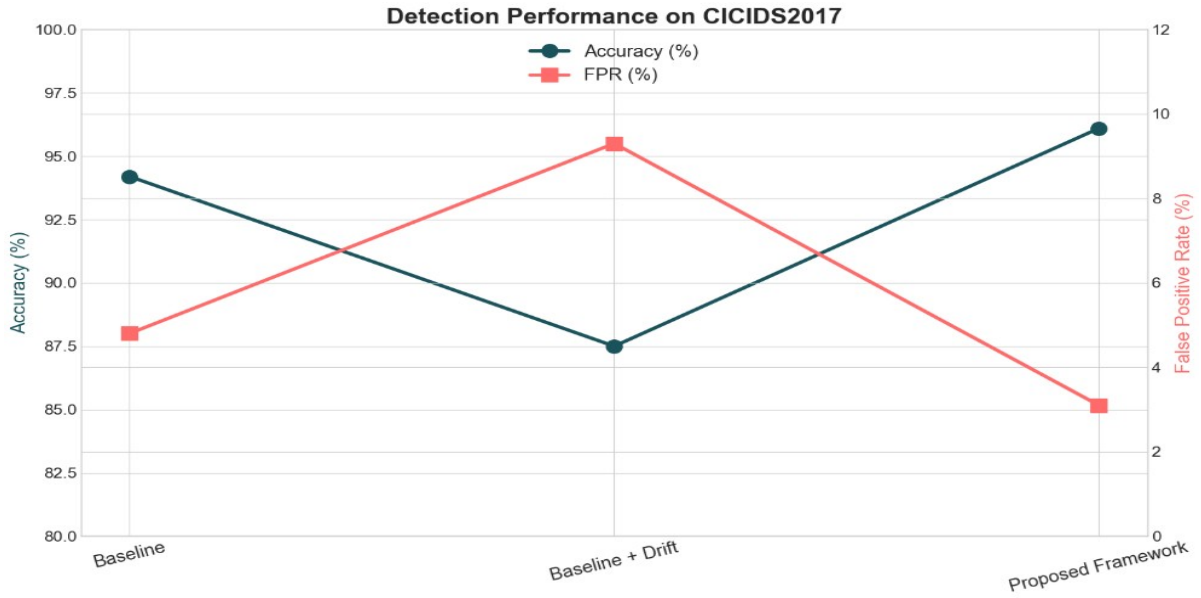

Figure 7: Detection performance graph.

5. Conclusion and future work

An adaptive stabilization framework was proposed to guarantee reliable analysis in drift, noise burst or adversarial environments. The architecture under consideration incorporates harmonization, deviation detection, adaptive parameter control, model consistency checks, and closed loop stabilization in a single pipeline. The outcomes demonstrated that this approach can greatly improve system functionality on all cases. During extreme drift, we saw a 30% boost in stability, recovery time drop threefold, adaptation overshoot diminish by 70%, and a solid model consistency score even under heavy distributional drift. In addition, detection performance was improved in real traffic with an accuracy of 96.1% and a false positive rate of 3.1%, confirming the relations that stabilization improves operational effectiveness.

The results show the usefulness controlled adaptation can confer on modern analysis. When there is no stabilization in a model, due to drift and adversarial perturbations, the model behaviour degrades quite fast. Moreover, the parameters start to oscillate excessively causing numerous false alarms. The framework as proposed reduces these effects by restricting adaptation, validating updates and monitoring of recovery dynamics. Consequently, the framework supports both short term stability and long term analytical coherence, making it suitable to run under a heavy load, in real time and adversarial settings.

Although it performs strongly, it has many limitations. The present strategy of employing heuristic policies is found to be effective, but it restricts the exploitation of long term temporal dependencies that are important for highly volatile data streams. The framework assumes drift events are independent, but real-world scenarios frequently involve correlated or cascading disturbances. The assessment also concentrated on network intrusion detection; wider validation in the financial, industrial, and IoT domains would help further validate the approach's generality.

The stabilization mechanism will undergo numerous extensions in future work. Having a model that predicts drift trajectory instead of merely reacting to it could be a promising addition. Introducing time-related attention mechanisms or sequence-aware meta-learning may aid in pre-computing destabilization events and adjusting parameters proactively.

An alternative path is to form adaptive policies which can change over time based on historical recovery patterns and the system's long-term stabilization. An important goal is to extend this framework to distributed and multi node environments for coordinated stabilization of

heterogeneous systems. In the end, future research will focus on explainability driven stabilization, in which the system provides interpretable justifications for adaptation decisions, to enhance transparency and trust in high stakes operational settings.

Declaration on Generative AI

During the preparation of this work, the authors used Microsoft Copilot and Google Gemini in order to: Grammar and spelling check. After using these tools/services, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] A. Kashtalian, L. Ścisło, R. Rucki, S. Lysenko, A. Sachenko, B. Savenko, O. Savenko, A. Nicheporuk, Control and Decision-Making in Deceptive Multi-Computer Systems Based on Previous Experience for Cybersecurity of Critical Infrastructure, *Applied Sciences* 15 (22) (2025) 12286. doi:10.3390/app152212286.

[2] D. Denysiuk, O. Savenko, S. Lysenko, B. Savenko, A. Nicheporuk, Detecting software implants using system decoys, *CEUR Workshop Proceedings* 3899 (2024) 277–287.

[3] O. Bokhonko, S. Lysenko, P. Gaj, Development of the social engineering attack models, *CEUR Workshop Proceedings* 3899 (2024) 288–303.

[4] Y. Lu, Efficient intrusion detection toward IoT networks using cloud–edge collaboration, *Computer Networks* 228 (2023) 109724. doi:10.1016/j.comnet.2023.109724.

[5] Y. Zhang, X. Zhe, Neural Network-Powered Intrusion Detection in Multi-Cloud and Fog Environments, *International Journal of Advanced Computer Science & Applications* 15 (2024). doi:10.14569/IJACSA.2024.0150625.

[6] M. Nachaat, Artificial intelligence and machine learning in cybersecurity: A deep dive into state-of-the-art techniques and future paradigms, *Knowledge and Information Systems* 67 (8) (2025) 6969–7055. doi:10.1007/s10115-025-02429-y.

[7] M. O. Okafor, Deep learning in cybersecurity: Enhancing threat detection and response, *World Journal of Advanced Research and Reviews* 24 (3) (2024) 1116–1132. doi:10.30574/wjarr.2024.24.3.3819.

[8] A. M. Dogo, B. N. Atanda, M. O. Oloyede, A Systematic Review of Concept Drift Approaches for Anomaly Detection in IoT Networks, *SN Computer Science* 7 (1) (2026) 18. doi:10.1007/s42979-025-04624-8.

[9] S. B. Mannapur, Understanding data drift and concept drift in machine learning systems, *International Journal of Scientific Research in Computer Science, Engineering and Information Technology* 11 (1) (2025) 318–330. doi:10.32628/CSEIT25111239.

[10] B. S. Purkayastha, M. M. Rahman, M. Shahpasand, Android malware detection using machine learning and neural network: A hybrid approach with federated learning, in: *Proceedings of the 2024 7th International Conference on Advanced Communication Technologies and Networking (CommNet)*, 2024, pp. 1–5. doi:10.1109/CommNet63022.2024.10793304.

[11] W. Hu, A deep analysis of nature-inspired and meta-heuristic algorithms for designing intrusion detection systems in cloud/edge and IoT: State-of-the-art techniques, challenges, and future directions, *Cluster Computing* 27 (7) (2024) 8789–8815. doi:10.1007/s10586-024-04385-8.

[12] E. D. V. Colter, R. M. A. Velásquez, L. Z. T. Espinoza, Comparative Systematic Review on Intelligent Threat Detection in Cloud Computing, *Proceedings of the Computational Methods in Systems and Software* (2025) 97–115. doi:10.1007/978-3-032-22091-2_8.

[13] M. Ozkan-Okay, A comprehensive survey: Evaluating the efficiency of artificial intelligence and machine learning techniques on cyber security solutions, *IEEE Access* 12 (2024) 12229–12256. doi:10.1109/ACCESS.2024.3355547.

[14] A. Alshuaibi, M. Almaayah, A. Ali, Machine learning for cybersecurity issues: A systematic review, *Security Challenges* 1 (2025) 2. doi:10.63180/jcsra.thestap.2025.1.4.

[15] M. U. Rashid, Hybrid android malware detection and classification using deep neural networks, *International Journal of Computational Intelligence Systems* 18 (1) (2025) 52. doi:10.1007/s44196-025-00783-x.

[16] S. Bashir, Hybrid machine learning model for malware analysis in android apps, *Pervasive and Mobile Computing* 97 (2024) 101859. doi:10.1016/j.pmcj.2023.101859.

[17] Z. B. Akhtar, T. A. Rawol, Enhancing cybersecurity through AI-powered security mechanisms, *IT Journal Research and Development* 9 (1) (2024) 50–67. doi:10.25299/itjrd.2024.16852.

[18] D. Kalla, Investigating the impact of heuristic algorithms on cyberthreat detection, in: *Proceedings of the 2024 2nd International Conference on Advances in Computation, Communication and Information Technology (ICAICCIT)*, 2024, pp. 450–455. doi:10.1109/ICAICCIT64383.2024.10912106.

[19] A. Petrovic, et al., Exploring metaheuristic optimized machine learning for software defect detection on natural language and classical datasets, *Mathematics* 12 (2024) 2918. doi:10.3390/math12182918.

[20] R. Fuller, et al., A novel hybrid machine learning approach for real-time ransomware detection using behavior-driven heuristic features, *Authorea Preprints* (2024). doi:10.22541/au.173161579.96102762/v1.

[21] A. Z. Abualkishik, N. Zikrillaeva, G. Gulyamova, Optimizing AI-Based Automated Security Patch Deployment in IoT Devices to Combat Zero-Day Exploits and Advanced Cyber Attacks, *Journal of Cybersecurity & Information Management* (2024). doi:10.54216/JCIM.130203.

[22] J. Liu, P. Cui, Data heterogeneity modeling for trustworthy machine learning, in: *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, Vol. 2, 2025, pp. 6086–6095. doi:10.1145/3711896.3736560.

[23] A. Yaseen, AI-driven threat detection and response: A paradigm shift in cybersecurity, *International Journal of Information and Cybersecurity* 7 (12) (2023) 25–43. doi:10.47672/ajce.2423.

[24] Y. Liu, Intelligent Analysis Methods for Multi-Channel Marketing Data Based on Anomaly Detection Algorithms, in: *Proceedings of the 2nd International Conference on Image Processing, Machine Learning, and Pattern Recognition*, 2025, pp. 198–206. doi:10.1145/3759928.3759963.

[25] D. Fayyad, Measuring the Human Risk Factor: Developing Predictive Models for Insider Threat Detection Using Behavioral Analytics, *International Journal of Humanities and Information Technology* 7 (4) (2025) 29–40. doi:10.21590/ijhit.07.04.03.

[26] U. Ahmed, et al., Signature-based intrusion detection using machine learning and deep learning approaches empowered with fuzzy clustering, *Scientific Reports* 15 (1) (2025) 1726. doi:10.1038/s41598-025-85866-7.

[27] R. Yu, et al., Ransomware detection using dynamic behavioral profiling: A novel approach for real-time threat mitigation, *Authorea Preprints* (2024). doi:10.36227/techrxiv.173047864.44215173/v1.

[28] S. Eisenwer, et al., Automated detection of ransomware using dynamic code sequence mapping, *Authorea Preprints* (2024). doi:10.36227/techrxiv.173014814.48823875/v1.

[29] S. F. Rabbani, Y. O. Rahat, M. K. Islam, AI-Driven Predictive Modeling for Detecting Suspicious Trading Patterns, Anomalous Order Activity, and Market Manipulation in US Equity Markets, *Journal of Economics, Finance and Accounting Studies* 8 (4) (2026) 13–27. doi:10.32996/jefas.2026.8.4.2.

[30] G. Shanks, et al., Innovative framework for ransomware detection using adaptive cryptographic behavior analysis, *Authorea Preprints* (2024). doi:10.22541/au.173230401.11413813/v1.

[31] O. Pomorova, O. Savenko, S. Lysenko, A. Kryshchuk, Multi-agent based approach for botnet detection in a corporate area network using fuzzy logic. Communications in Computer and Information Science 370 (2013) 243–254.