

CD-AHAL: A CONCEPT DRIFT-AWARE HYBRID ACTIVE LEARNING FRAMEWORK FOR NETWORK INTRUSION DETECTION

PHUOC ANH DUNG NGUYEN¹, VIET HUNG NGUYEN¹, VAN THANG TRAN¹,
VAN QUY HOANG², TAT THANG NGUYEN³, THIEN HUY LE⁴, HUONG BUI^{1,*}

¹*Faculty of Information Technology, HUTECH University, 475A Dien Bien Phu Street,
Thanh My Tay Ward, Ho Chi Minh City, Viet Nam*

²*Thuy Loi University (TLU), 175 Tay Son Street, Kim Lien Ward, Ha Noi, Viet Nam*

³*GEO IT Center, VNPT Information Technology Company, VNPT Group, VNPT Tower,
57 Huynh Thuc Khang Street, Lang Ward, Ha Noi, Viet Nam*

⁴*Hoa Sen College, Lot 11, Street No. 5, Quang Trung Software City, Trung My Tay Ward,
Ho Chi Minh City, Viet Nam*



Abstract. This paper proposes a solution to the challenges posed by the proliferation of IoT devices, which has expanded cyberattack surfaces characterized by continuous changes over time (Concept Drift). Traditional Network Intrusion Detection Systems (NIDS) often suffer from severe performance degradation when facing novel attacks due to a lack of adaptability. This paper introduces CD-AHAL, a hybrid security framework that integrates a CNN–BiGRU–Attention deep learning architecture with an Uncertainty-aware Active Learning strategy, employing an automatic labeling mechanism via a simulated Oracle. The system is evaluated through a two-stage process: (1) Offline training to establish a foundational knowledge base; and (2) Online evaluation on a synthetic data stream exhibiting concept drift. To demonstrate comprehensive effectiveness, we compare CD-AHAL against three baseline architectural variants CNN-GRU, CNN-Attention, and GRU-Attention operating in a static (non-updating) mode. Experimental results demonstrate that while static models suffer from accuracy degradation in changing environments, CD-AHAL is capable of autonomously detecting drift and rapidly recovering performance, thereby maintaining superior average accuracy and F1-score.

Keywords. Active learning, CD-AHAL, class imbalance, CNN–GRU–Attention, concept drift, IoT security, network intrusion detection.

1. INTRODUCTION

In the era of the Internet of Things (IoT), the heterogeneity and complexity of interconnected devices have created highly dynamic and vulnerable network environments. Although Network Intrusion Detection Systems (NIDS) serve as an essential line of defense, traditional deep learning models often suffer from poor generalization under changing traffic conditions [1, 2]. Two key challenges are severe class imbalance, which causes rare attacks to be overlooked, and concept drift, where evolving attack patterns make previously learned knowledge obsolete. Recent studies further show that AI-powered NIDS can be vulnerable

E-mail addresses: anhdungnguyen955@gmail.com (P.A.D. Nguyen); nviethung23@gmail.com (V.H. Nguyen); tranthang2442004@gmail.com (V.T. Tran); hoangvanquy@tlu.edu.vn (V.Q. Hoang); nguyentatthang@vnpt.vn (T.T. Nguyen); lethienhuy0804@gmail.com (T.H. Le); bd.huong@hutech.edu.vn (H. Bui).

to adversarial attacks, reducing detection reliability under hostile traffic conditions [3].

A static model, even if accurate in laboratory settings, can quickly degrade when deployed on real-world streaming data. Recent defense-oriented studies have emphasized adaptive robustness and evasion mitigation for deep learning-based intrusion detection systems [4, 5]. To address these limitations, we propose CD-AHAL (concept drift-aware hybrid active learning), a framework that combines a CNN-BiGRU-Attention backbone with Active Learning [6, 7]. By using MC Dropout [8] to estimate predictive uncertainty, CD-AHAL selects informative samples for retraining and enables rapid recovery after concept drift events.

Scientifically, this study bridges uncertainty estimation and continuous learning in cybersecurity. Instead of relying only on reactive error-based drift detection, CD-AHAL uses Entropy as a proactive early-warning signal. Practically, the framework reduces retraining and annotation costs through simulated Oracle labeling, making it suitable for low-latency IoT NIDS deployment. Although this study mainly focuses on concept drift, the proposed adaptive mechanism also provides a foundation for future adversarial robustness extensions [3–5].

The main contributions of this study are summarized as follows:

1. **Optimized Hybrid Architecture:** We integrate CNN, Bi-GRU, and Attention mechanisms to improve spatial feature extraction, temporal modeling, and critical signal prioritization for attack detection.
2. **Adaptive Active Learning Mechanism:** We employ MC Dropout to quantify uncertainty, detect drift, and trigger cost-effective retraining with reduced labeling effort.
3. **Static vs. Dynamic Evaluation:** We compare CD-AHAL with conventional static baselines across multiple architectural variants, validating the importance of Active Learning for real-world deployment.

The remainder of this paper is organized as follows. Section 2 reviews related work and formulates the research questions. Section 3 presents the CD-AHAL architecture. Section 4 describes the experimental setup and results. Section 5 concludes the paper and outlines future work.

2. RELATED WORK

CNN-based models have been widely used in NIDS due to their efficient local feature extraction and low computational cost [9]. However, they are limited in capturing long-term temporal dependencies in complex attacks. Recurrent models such as LSTM and GRU address this issue by modeling temporal patterns [10, 11]. Attention-based and Transformer models have also shown strong performance, but their computational overhead limits their suitability for edge or near real-time deployment [1, 2]. Therefore, a lightweight hybrid architecture that balances detection performance and operational cost remains necessary.

Class imbalance is another persistent challenge in NIDS, often biasing models toward majority benign traffic. Although SMOTE [12] is effective in offline training, applying it directly to streaming data may distort the real-world traffic distribution. In addition, NIDS models frequently degrade under domain shifts and concept drift caused by emerging attack variants [13]. Beyond natural drift, adversarial manipulation is also a growing threat, as adversarial samples can alter traffic representations and mislead deep learning-based detectors [3]. Recent studies have explored adaptive defense strategies and autoencoder-based mitigation against evasion attacks [4, 5], but they do not fully address online adaptation under evolving streaming traffic.

To address distribution shifts, several drift detection and online updating strategies have

been proposed, but they still have limitations:

1. *Continuous Online Deep Learning (ODL)* [14]: ODL enables rapid adaptation but requires frequent updates, leading to high computational cost and possible catastrophic forgetting.
2. *Reactive Drift Detectors* [15]: Methods such as ADWIN rely on error-rate changes, meaning they react only after misclassifications occur.
3. *Inefficient Active Learning* [16]: Although Active Learning reduces labeling cost [6], many NIDS studies still rely on basic uncertainty-based sampling, which may waste labeling budgets during abrupt drift.

Based on these limitations, this study addresses the following research questions:

- *RQ1*: How can concept drift be detected proactively before significant classification errors occur?
- *RQ2*: How can Active Learning update the model with minimal labeling cost while maintaining detection accuracy?

Active Learning is effective for reducing labeling effort by selecting high-uncertainty samples [6]. However, the integration of MC Dropout-based uncertainty estimation [8] with a lightweight drift-aware NIDS remains underexplored. Although prior works emphasize adversarial defense and evasion mitigation in AI-based NIDS [3–5], uncertainty-guided active learning for online concept-drift adaptation is still insufficiently investigated. Therefore, this study uses Entropy via MC Dropout as an early warning signal to trigger retraining, balancing accuracy and computational cost [17].

3. METHODOLOGY

3.1. Overall architectural model

The CD-AHAL framework is designed as a continuous learning pipeline that seamlessly connects static offline initialization with dynamic online adaptation. The overall architecture is built upon a hybrid deep learning backbone, specifically a CNN-BiGRU-Attention model. The rationale behind this integration is to balance robust feature extraction with low inference latency. The CNN is highly efficient at capturing local spatial correlations from standardized network flows, while the Attention mechanism dynamically assigns weights to critical features, ensuring that subtle anomalies from rare attacks are not overshadowed by dominant benign traffic.

The proposed system is divided into four tightly integrated functional blocks (see Figure 1):

- **Block 1: Frozen Preprocessing Pipeline.** *Function*: Standardizes raw network traffic into a unified feature space. *Contribution/Rationale*: When transitioning from the offline training phase to the online streaming phase, all normalization parameters are strictly frozen. This prevents data leakage and accurately simulates real-world operational constraints where future data distributions are unknown.
- **Block 2: CNN-BiGRU-Attention Feature Extractor & Classifier.** *Function*: The core detection engine. The 1D-CNN extracts local features, which are fed into a Bi-GRU for temporal modeling, and then passed to the Adaptive Attention Mechanism. *Contribution/Rationale*: The CNN provides lightweight processing suitable for high-speed network traffic. The Attention Mechanism serves as a targeted focal point, addressing class imbalance by selectively prioritizing minority attack classes and filtering out irrelevant background noise.

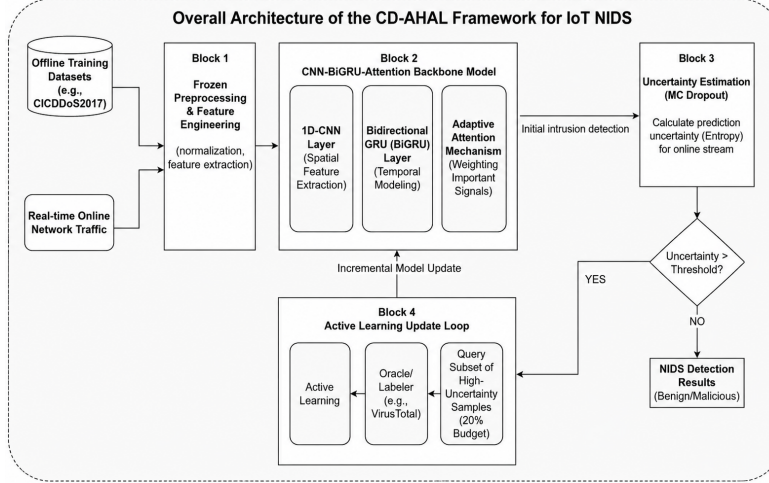


Figure 1: Overall architecture of the CD-AHAL framework

- **Block 3: Uncertainty Estimation Module (MC Dropout).** *Function:* Quantifies the model’s predictive uncertainty (Entropy) for every incoming data sample utilizing MC Dropout [8]. *Contribution/Rationale:* Instead of waiting for misclassifications to occur, the framework uses MC Dropout to execute stochastic forward passes. A sudden spike in the calculated Entropy serves as a proactive, early-warning signal that the data distribution has shifted (Concept Drift). This is far more effective than traditional error-rate monitors like ADWIN [15].
- **Block 4: Active Learning & Oracle Labeling.** *Function:* Manages the retraining budget and updates the model dynamically based on Active Learning principles [6, 7]. *Contribution/Rationale:* When a drift signal is triggered by Block 3, this module selects only the samples with the highest uncertainty (Entropy) and queries a simulated Oracle for ground-truth labels. By strictly limiting the labeling budget (e.g., to 20% of the batch), this block ensures rapid fine-tuning without exhausting computational resources or human-annotation effort.

3.2. Investigated model architectures

1. The proposed CD-AHAL framework is the integration of a CNN + Bi-GRU + attention architecture. This is the only model equipped with a drift-aware active learning mechanism in Phase 2.

2. CNN-GRU (baseline 1): The attention module is removed. The model operates under a static (non-adaptive) setting in Phase 2. The objective of this model is to isolate the effect of attention and quantify performance degradation when no adaptation mechanism is available.

3. CNN-Attention (baseline 2): The Bi-GRU module is removed. The model operates in a static setting. The objective of this model is to assess the importance of temporal dependency modeling (sequence learning).

4. GRU-Attention (baseline 3): The CNN module is removed. The model operates in a static setting. The objective of this model is to examine the role of CNNs in denoising and local feature extraction.

3.3. Multi-source dataset design and experimental setup

To evaluate the CD-AHAL framework under practical conditions, we design a two-phase experimental protocol that simulates a transition from conventional IoT networks to critical

infrastructure environments.

Phase 1 is offline training. In this phase, we use CICDDoS2017 [18] and CICIoT2023 [19] as the foundation data. This combination helps the model capture the basic features of DDoS attacks and the behavior of IoT devices.

Phase 2 is composite drift evaluation. To simulate concept drift in the data stream, we build a mixed test stream from CSE-CIC-IDS2018 [18], CICDDoS2019 [20], CICDarknet2020 [21], and CIRA-CIC-DoHBrw-2020 (L1-DoH-NonDoH) [22]. This data stream forces the model to face attack types it has never encountered during training.

All datasets are standardized into a shared feature space to ensure consistent input representations across sources. The dataset composition and distribution are summarized in Table 1. After the unified preprocessing stage, the offline training dataset aggregates to exactly 4,800,000 samples, while the online composite drift stream consists of 1,277,823 samples. Both datasets are mapped into a shared 33-dimensional feature space to ensure consistency during model inference. To ensure feasibility on resource-constrained experimental hardware, we removed redundant samples and aggregated every 10 consecutive network records into a single sequential block. This approach effectively reduced the data scale tenfold from 1,277,823 raw samples to 127,488 sequences significantly optimizing computational efficiency and memory usage while preserving essential temporal features across 249 processed batches (512 sequences per batch).

Table 1: Datasets used in this study

Dataset	Role	Domain/Focus	Sample	Columns
CICDDoS2017	Training	DDoS-centric traffic	800,000	32
CICIoT2023	Training	IoT traffic with emerging attack variants	4,000,000	14
CSE-CIC-IDS2018	Drift testing	Broad enterprise-scale cyberattacks	50,000	15
CICDDoS2019	Drift testing	Large-scale and diverse DDoS scenarios	254,528	32
CICDarknet2020	Drift testing	Anonymized/darknet-related attacks	923,295	8
L1-DoH-NonDoH	Drift testing	Encrypted DNS (DoH vs. non-DoH) traffic	50,000	8

3.4. Data preprocessing pipeline

To ensure an objective evaluation across heterogeneous data sources, we establish a rigorous two-stage preprocessing workflow as follows:

Offline phase (training): Traffic records from CICDDoS2017 and CICIoT2023 are cleaned and aligned to derive a shared feature space. We address class imbalance by combining SMOTE [12] with a class-weighted loss function, enabling the initial model to learn rare attack patterns more effectively (see Figure 2).

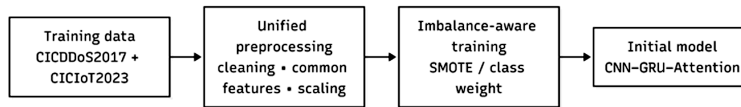


Figure 2: Offline training phase

Online adaptation phase: When deployed on the composite drift stream, we apply the Frozen Preprocessing mechanism. Specifically, all new input data is standardized using parameters fixed from the offline phase. This prevents data leakage and ensures realism when the system must process previously unseen data (see Figure 3).

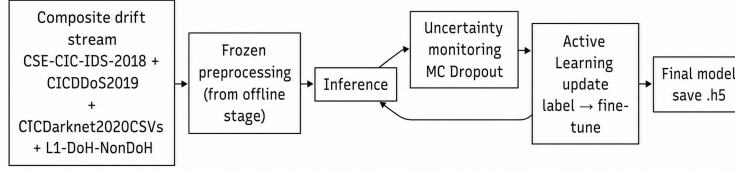


Figure 3: Online adaptation phase

SMOTE is replaced with Active Learning [6]. Synthetic oversampling methods such as SMOTE are not applied in the online stream to avoid distorting the true operational distribution. Instead, drift is handled via an active learning mechanism that selectively requests labels for highly uncertain samples, where uncertainty is estimated using MC Dropout [8] and then uses these informative instances to update the model.

3.5. Adaptation mechanism uncertainty-based active learning

To cope with concept drift in streaming environments, CD-AHAL incorporates an online adaptation strategy guided by predictive uncertainty (uncertainty-guided adaptation).

Uncertainty formulation: We employ MC Dropout [8] to approximate Bayesian inference without incurring substantial computational overhead. At inference time, for each input sample x , the model performs T stochastic forward passes by keeping dropout active. Predictive uncertainty is then quantified using the entropy H of the mean predictive probability distribution

$$p(y = c|x) \approx \frac{1}{T} \sum_{t=1}^T p(y = c|x, \theta_t), \quad (1)$$

$$H(x) = - \sum_{c=1}^C p(y = c|x) \log p(y = c|x). \quad (2)$$

In these equations, C denotes the number of classification classes, while θ_t represents the model parameters sampled during the t -th stochastic forward pass induced by random dropout. A larger entropy value $H(x)$ suggests that the corresponding input sample x is more likely to be novel or out-of-distribution (OOD).

Batch-level entropy aggregation:

To evaluate the overall uncertainty of an incoming data stream at a macro level, we aggregate the individual predictive entropies into a batch-level metric. For a given streaming batch B containing N instances, the batch-level entropy, denoted as $H_{batch}(B)$, is defined as the mean entropy of all samples within that batch

$$H_{batch}(B) = \frac{1}{N} \sum_{x_i \in B} H(x_i). \quad (3)$$

Justification of the entropy threshold and drift signal:

It is important to contextualize why an aggregated H_{batch} value (e.g., 0.349 observed during our streaming evaluation) is considered a strong drift signal, despite being seemingly low compared to the theoretical maximum entropy bounds (e.g., $\log_2(C)$). In NIDS environments, benign traffic constitutes the vast majority of the data stream. A well-trained, highly confident model typically yields an extremely low baseline entropy (approaching 0)

for normal operations. Therefore, any sudden, aggregated spike in H_{batch} (such as 0.349) represents a massive relative deviation from the stable baseline, indicating significant model confusion.

Furthermore, utilizing the aggregated H_{batch} rather than the individual $H(x)$ allows CD-AHAL to distinguish between genuine concept drift and transient noise. Isolated data anomalies or transient noise may cause a high $H(x)$ for a few individual samples, but their mathematical effect is heavily diluted when averaged over a sufficiently large batch size N . Consequently, a significant and sustained increase in H_{batch} strictly correlates with a macro-level domain shift or the emergence of novel attack variants (Concept Drift), warranting an immediate active learning update.

Labeling budget strategy: In Phase 2, once drift is detected, the system enforces a constrained labeling budget to optimize resource usage and maintain timely responsiveness:

Fixed budget (B): Instead of requiring labeling for all new data, we limit label queries from Oracle to 20% of the number of samples in each batch of data detected as drifting.

Simulated Oracle mechanism: In this experimental framework, the role of Oracle is simulated by retrieving actual labels (ground truth) available in the test datasets. This design simulates the real-world process where the system automatically sends suspicious samples to third-party services (such as VirusTotal, AbuseIPDB) or uses a stronger Teacher Model for identification, rather than relying on manual operations by human experts.

Sampling process: This process includes two steps as follows.

Step 1 (Uncertainty sampling): The system calculates the Entropy index from MC Dropout predictions. Samples with the highest Entropy values (indicating the model’s “confusion”) are prioritized for sending queries to the Oracle to obtain the most accurate labels for updating weights.

Step 2 (High-confidence Pseudo-labeling): For the remaining samples that are not selected for labeling, if the model has a very high prediction confidence (Confidence > 0.95), the system will automatically assign a “pseudo-label” based on the current prediction. These samples are added to the supplementary training set to reinforce knowledge without incurring labor costs.

Temporal dependency modeling (Bi-GRU): To capture the evolving nature of modern attacks (e.g., long-duration or slowly varying DDoS campaigns), the CNN-extracted features are fed into a bidirectional GRU (Bi-GRU). Unlike purely feed-forward architectures, Bi-GRU leverages both past and future context within the traffic sequence, which improves temporal robustness and helps stabilize predictions under time-evolving concept drift.

Adaptive attention mechanism: This is the key component for addressing data imbalance. The Attention Mechanism allows the model to automatically assign higher weights to time points containing important information and ignore segments of noise. As a result, weak or rare attack signals (minority classes) are not “drowned out” by the vast sea of benign traffic.

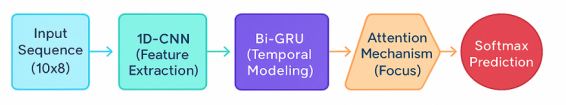


Figure 4: CNN-BiGRU-Attention architecture

The output of the attention block is passed through fully connected (dense) layers to produce the final feature vector, which is then used for downstream classification tasks. The detailed internal structure of this feature extractor and classifier is depicted in Figure 4.

3.6. Two-phase detection strategy

To balance real-time alerting requirements with the need for fine-grained analysis to support digital forensics, CD-AHAL operates under a two-stage pipeline and reuses a shared feature extractor to reduce computational overhead:

Phase 1: Rapid Screening (Binary Detection): The top priority of NIDS is early detection. Therefore, the initial model is configured with the Binary Cross-Entropy loss function to perform the binary classification task: Normal (Benign) versus Attack. This stage helps to quickly remove most of the clean traffic to reduce the load on the system and provide immediate alerts when there are signs of abnormality.

Phase 2: Fine-grained Identification (multiclass classification): Once a flow is flagged as malicious, the system switches to multiclass classification using categorical cross-entropy. At this stage, the model assigns the traffic instance to a specific attack category (e.g., DDoS, PortScan, Botnet, WebAttack, etc.) according to the predefined label space.

This hierarchical design improves overall accuracy by allowing each stage to specialize in a distinct objective, while also maintaining deployment feasibility on resource-constrained edge devices.

3.7. Handling concept drift via uncertainty-based active learning

To ensure the CD-AHAL model can operate sustainably in a real-world environment where new attack variants continuously emerge we established an online adaptation process based on two principles:

1. Preserving data integrity: When transitioning from offline training to streaming deployment, the entire preprocessing pipeline is frozen. Normalization parameters estimated in the offline phase are reused for all incoming samples, and no synthetic data generation techniques (e.g., SMOTE) are applied to the online stream. This design prevents data leakage and more faithfully reflects operational constraints.

2. Uncertainty-guided monitoring and updates: Instead of labeling all incoming data (which is costly and impractical), CD-AHAL leverages MC Dropout to estimate predictive uncertainty for each instance, quantified by the mean entropy computed over $T = 10\text{--}50$ stochastic forward passes.

Drift detection: A sudden increase in entropy serves as an early-warning signal indicating potential concept drift or domain shift (e.g., the emergence of new attack variants or abrupt changes in traffic behavior). Upon detecting this signal, the system triggers the active labeling procedure.

Labeling strategy: The system activates the Active Learning loop, only querying the Oracle for labels of the most uncertain samples. Using the Oracle instead of humans allows the system to operate fully automatically. In a real-world deployment environment, this Oracle would be APIs (VirusTotal, AbuseIPDB) or heavyweight Teacher Models. In this study, we use the original labels of the data to simulate the absolute accuracy of the Oracle's response.

Fine-tuning: The model performs rapid incremental fine-tuning on the newly labeled, small-scale dataset, allowing performance recovery without retraining from scratch.

3.8. Training configuration

All models are trained under consistent settings to ensure fair comparability between the baselines and the proposed CD-AHAL architecture. We adopt binary cross-entropy for Phase 1 and categorical cross-entropy for Phase 2, together with class weighting to emphasize minority attack classes.

We employ Adam/AdamW optimizers, with the initial learning rate tuned within a narrow range (typically 1×10^{-3} to 5×10^{-4}) and reduced via a scheduler when the validation loss plateaus. To mitigate overfitting, dropout (rate 0.3–0.5) is applied across the CNN, GRU, and dense layers. Early stopping is also enabled, monitoring validation loss with a patience of 5–10 epochs.

Table 2: Training hyperparameters

Hyperparameter	Value/Setting
Optimizer	Adam / AdamW
Initial learning rate	$\sim 1e-3$ to $5 \times 1e-4$ (with LR scheduler if needed)
Batch size	256–1024 (depending on hardware)
Max epochs	30–100
Early stopping	Monitor val_loss, patience 5–10 epochs
Dropout	0.3–0.5 (CNN/GRU/Dense)
Loss (Phase 1)	Binary Cross-Entropy
Loss (Phase 2)	Categorical Cross-Entropy
Imbalance strategy (offline)	SMOTE and/or class-weighted loss
Online setting	Frozen preprocessing, no SMOTE

SMOTE is applied exclusively in the offline phase to balance the multi-source training data (CICDDoS2017 + CICIoT2023) for constructing the initial model (as detailed in Table 2).

Performance is evaluated using metrics sensitive to imbalance, including Accuracy, Precision, Recall, F1-score, AUC-ROC/AUC-PR, and confusion matrix.

4. EXPERIMENTAL RESULTS

4.1. Experimental environment

Table 3 reports the hardware and software configuration used in our experiments. The system is deployed on a personal workstation to emulate an edge-computing setting.

Table 3: Experimental environment configuration

Item	Component	Detailed specification
Hardware	CPU	Intel Core i7-12700H (14 cores, 20 threads, up to 4.70 GHz)
	GPU	NVIDIA GeForce GTX 3050Ti (4GB GDDR6 VRAM)
	RAM	32GB DDR4 3200MHz
Software	OS	Windows 11 Home Single Language (64-bit)
	Framework	TensorFlow 2.10.0 (GPU support), Keras 2.10
	Language	Python 3.9.13 (Libraries: NumPy 1.23.5, Pandas 1.5.3, Scikit-learn 1.0.2)

4.2. Offline training results for Phase 1

To evaluate adaptability, we conducted a pruning study to demonstrate the superiority of the CNN-GRU-Attention hybrid architecture over other variants on the static dataset.

Table 4: Phase 1 binary training comparison results

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
CD-AHAL (Proposed)	98.53	98.90	99.20	99.05
CNN-GRU (Baseline)	98.66	98.66	98.66	98.66
GRU-Attention	98.72	98.72	98.72	98.72
CNN-Attention	98.73	98.72	98.73	98.73

As shown in Table 4, CNN-Attention achieves the highest accuracy of 98.73%. CD-AHAL obtains a slightly lower accuracy of 98.53%, but achieves higher recall of 99.20% and F1-score of 99.05% compared with the other models. These results indicate that CD-AHAL has strong attack detection capability, particularly in identifying malicious samples.

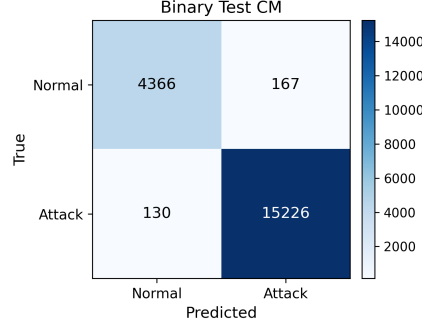


Figure 5: CD-AHAL binary confusion matrix

The binary confusion matrix in Figure 5 demonstrates the effectiveness of the CD-AHAL model, with an extremely low misclassification rate between the Normal and DDoS classes in the offline test dataset.

Table 5: Phase 1 multiclass training results comparison

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
CD-AHAL (Proposed)	77.00	78.00	77.00	78.00
CNN-GRU (Baseline)	78.57	80.53	78.57	78.64
CNN-Attention	78.47	81.06	78.47	78.30
GRU-Attention	78.72	82.02	78.72	78.31

The performance of the baseline and proposed models is shown in Table 5.

Baseline performance: The baseline models exhibit stable performance in this phase. In particular, GRU-Attention achieves the strongest results across most metrics, with an accuracy of 78.72% and a precision of 82.02%.

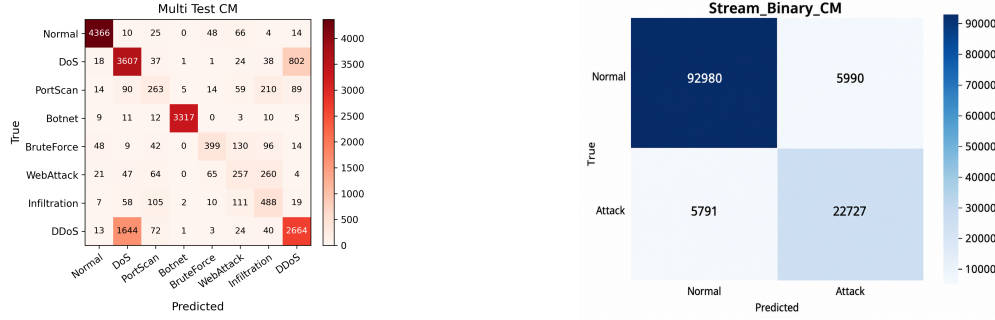
CNN-GRU and CNN-Attention follow closely, with accuracies of 78.57% and 78.47%, respectively. This result indicates that hybrid architectures incorporating attention are effective in extracting discriminative features under a static attack distribution.

Proposed model performance (CD-AHAL):

During Phase 1 initialization, CD-AHAL attains an accuracy of 77.00% and an F1-score of 78.00%. Although CD-AHAL is approximately 1.7 percentage points lower than the best-performing baseline (GRU-Attention) in overall accuracy, the gap is modest. The relatively balanced precision (78%) and recall (77%) suggest that the model does not overly bias toward either false positives or false negatives. Figure 6(a) presents the detailed confusion matrix for the Phase 1 multiclass evaluation, while Figure 6(b) shows the binary confusion matrix for the Phase 2 drift evaluation.

4.3. Phase 2 drift evaluation results

This is the most critical stage of the study. We deploy the models on a continuous data stream containing distribution shift events (Concept Drift).



(a) Phase 1 multiclass confusion matrix

(b) Binary confusion matrix for online CD-AHAL

Figure 6: Confusion matrices for multiclass and binary CD-AHAL evaluation

Table 6: Phase 2 binary training results comparison

Model architecture	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
CD-AHAL (Proposed)	90.76	90.78	90.76	90.77
CNN-GRU (Baseline)	77.79	73.05	77.79	72.05
CNN-Attention	77.56	71.86	77.56	70.34
GRU-Attention	77.56	71.86	77.56	70.34

As shown in Table 6, while the baseline models (CNN-GRU, GRU-Attention, and CNN-Attention) achieve an accuracy of around 77%, CD-AHAL reaches 90.76%.

The F1-score of CD-AHAL (90.77%) is over 18% higher than that of the baseline model (72.05%). This demonstrates that the CD-AHAL architecture handles the complex features of each attack type significantly better than the other architectures. The specific true/false positive and negative distributions for this stream are shown in Figure 6(b).

Drift event analysis: As visualized in the accuracy trajectory in Figure 8, the experimental logs record three major drift events:

Event 1 (Initial warm-up and abrupt shock, Batches 0–9):

Right at the beginning of the stream, the model encounters a sharp distributional change, causing accuracy to drop to its absolute lowest point of $\sim 73\%$ (Batch 3). However, the Active Learning mechanism proved immediately effective. Driven by rapid updates from high-uncertainty samples, the accuracy recovered strongly and rapidly, reaching $\sim 86\%$ by Batch 5 and surging to over 90% by Batch 9 (an impressive increase of +17% after just a few adaptation cycles). This demonstrates the system’s rapid “self-healing” capability during the turbulent initial phase.

Event 2 (Breakthrough adaptation, Batches 25–30):

Approaching Batch 29, the accuracy fluctuates around 85–87%. When the system detects a strong drift signal characterized by high entropy, the active labeling procedure is triggered again (visible as red markers around Batch 30). As a result, the model achieved a significant leap in performance, immediately boosting accuracy back to the 90–92% range. This demonstrates that CD-AHAL’s entropy-based sampling strategy not only helps the model recover but also enables it to learn new knowledge to achieve significantly higher performance than its previous fluctuating state.

Event 3 (Stability under a noisy regime, Batches 230–249):

Toward the end of the experimental cycle, the model encounters a dense series of consecutive drift episodes (composite drift), occurring rapidly between Batches 233 and 245. Despite these repeated distribution shifts (indicated by the dense cluster of drift markers),

the accuracy not only avoids degradation but remains highly stable at a baseline of $\sim 90\%$. Notably, the system even hits peak accuracies of up to 94% immediately after these updates. This finding strongly supports the long-term robustness and fatigue resistance of the CD-AHAL architecture in prolonged deployments.

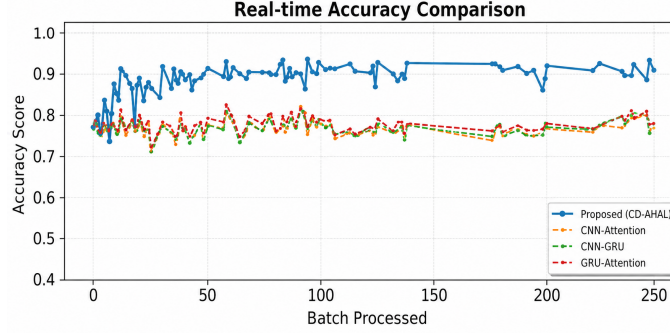


Figure 7: Visual illustration of drift events

Multiclass performance analysis:

Table 7: Phase 2 multiclass training results comparison

Model architecture	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
CD-AHAL (Proposed)	77.99	78.09	77.99	77.33
CNN-GRU (Baseline)	75.20	63.98	75.20	67.76
CNN-Attention	76.13	62.49	76.13	67.70
GRU-Attention	76.13	62.49	76.13	67.70

The performance of the proposed CD-AHAL model and the baseline models is shown in Table 7. The proposed CD-AHAL model achieves the highest accuracy of 77.99%, outperforming CNN-Attention and GRU-Attention, which both plateau at 76.13%, by 1.86 percentage points. CD-AHAL also attains the highest precision of 78.09%, substantially exceeding the remaining baseline models, whose precision values range from approximately 62% to 64%. Furthermore, CD-AHAL achieves an F1-score of 77.33%, which is markedly higher than the baseline group, whose F1-scores remain around 67.7%. This high precision indicates that CD-AHAL produces fewer false alarms than the competing approaches in the multiclass detection scenario.

4.4. Comparison with state-of-the-art methods

To evaluate robustness under streaming concept drift, we compare CD-AHAL with representative baselines from different paradigms: Adaptive Random Forest (ARF) [23] as a strong ensemble stream learner, Online Deep Learning (ODL) [14] as a continuous-update deep baseline, CNN-BiGRU + ADWIN [15] as a drift detector-triggered adaptation approach, and Periodic Retraining as a fixed-schedule update strategy. All methods follow the same prequential evaluation protocol on identical streaming batches and feature settings.

Figure 8 shows the accuracy trajectory over the 249 streaming batches. ARF consistently maintains the highest accuracy trend (approximately 0.94–0.95), while ODL stays slightly lower but stable (around 0.90–0.92). CD-AHAL remains highly competitive (averaging around 0.90–0.92 after overcoming the initial distribution shock) and exhibits rapid recovery after distribution shifts. The red cross (‘x’) markers indicate recovery points following drift-aware adaptation, highlighting that CD-AHAL can regain and sustain high performance quickly without continuous updates. In stark contrast, CNN-BiGRU + ADWIN and

Periodic Retraining suffer severe degradation over time, with their accuracies plummeting to the 0.70–0.80 range. This reflects their limited responsiveness to complex and continuous drift under the same streaming conditions.

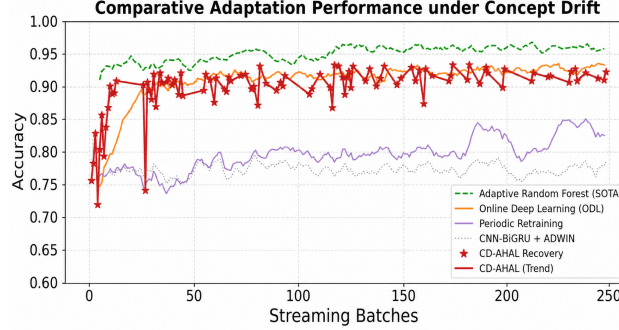


Figure 8: Performance comparison with state-of-the-art methods

Table 8: Performance metrics comparison (Accuracy/Precision/Recall/F1-score) and adaptation cost (Update Count/Update Rate/Total Times/Avg. Cost per Batch) across baselines.

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Update Count	Update Rate (%)	Total Time (ms)	Avg. Cost/Batch (ms)
CD-AHAL	90.76	90.78	90.76	90.77	97	38.9	74,630	300
ARF	95.00	99.26	78.07	87.28	Continuous	100	164,806	667
ODL	91.14	90.74	67.14	76.32	249	100	42,418	171
CNN-BiGRU + ADWIN	78.00	53.32	13.03	20.83	0	0	0	0
Periodic Retraining	80.20	71.91	19.27	29.69	12	4.8	2,060	8

It can be shown in Table 8 that:

Detection Quality (Recall & F1-Score): While ARF achieves the highest Accuracy (95.00%), its Recall (78.07%) is significantly lower than CD-AHAL (90.76%). This indicates that ARF tends to miss minority attack instances. CD-AHAL delivers the best balance with the highest F1-Score (90.77%), ensuring robust security and high detection coverage in dynamic environments.

Computational Cost (Total Time & Avg. Cost): The last two columns reveal the practical feasibility of each model regarding deployment on edge devices.

ARF & ODL: Due to a 100% update rate, ARF incurs a massive total time of 164,806 ms, translating to an average cost of 667 ms per batch. This high latency (>0.5s) can create a bottleneck for real-time traffic processing at the gateway level.

CD-AHAL: By triggering updates only 97 times (38.9%), our model maintains a total adaptation time of 74,630 ms. Consequently, the amortized average cost per batch is only ~300 ms, which is significantly below the sub-second threshold.

Conclusion: CD-AHAL achieves approximately a 2.2x reduction in total latency compared to ARF while maintaining superior Recall and F1-Score. This confirms that CD-AHAL is the most suitable solution for resource-constrained edge IoT environments where both high security and low-latency response are critical requirements.

4.5. Analysis of computational cost and actual latency

Deployment feasibility on edge IoT devices depends heavily on the model update overhead. Figure 10 compares the number of model updates across methods. While ODL [14] and ARF [23] perform continuous updates on 100% of batches (249 updates), CD-AHAL triggers updates on only 38.9% of batches (97 updates). This corresponds to an approximately 61% reduction in retraining-related computational cost, while still maintaining competitive accuracy.

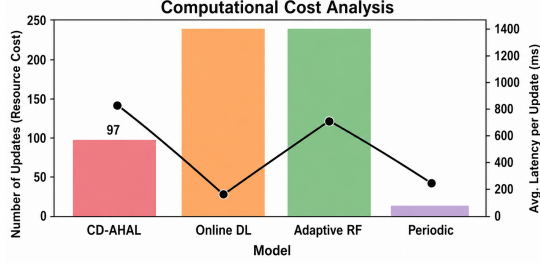


Figure 9: Comparison of model update cost and computational efficiency

In addition, we conduct an in-depth latency analysis of the adaptation pipeline in Figure 10. The average time per adaptation cycle (detection + labeling + training) is ~ 769 ms, with a median of approximately 760 ms. High-latency outliers (>2000 ms) occur only rarely and are mainly observed during the initial warm-up stage (Batches 0–1). With sub-second processing latency, CD-AHAL satisfies soft real-time requirements on resource-constrained IoT gateways.

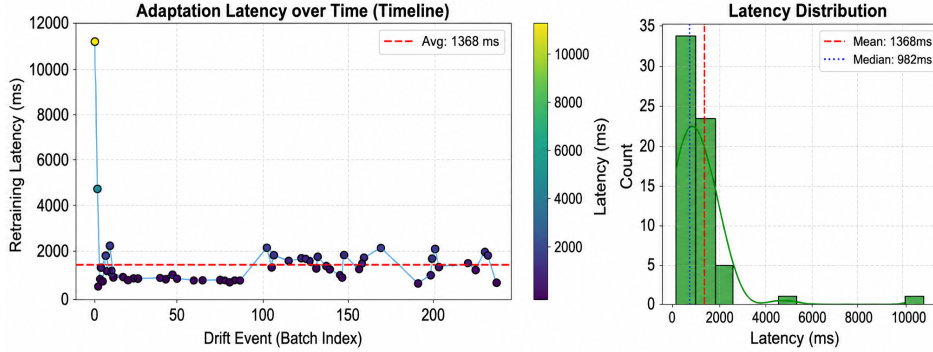


Figure 10: Latency distribution and temporal stability analysis of CD-AHAL

5. CONCLUSIONS

This paper proposed and conducted a comprehensive evaluation of CD-AHAL, a hybrid network intrusion detection framework optimized for dynamic IoT environments. By combining the feature extraction capability of a CNN–BiGRU–Attention backbone with an uncertainty-guided active learning strategy, the proposed approach effectively addresses the dual challenges of concept drift and class imbalance.

Extensive experiments demonstrate that CD-AHAL provides a balanced and practical solution for IoT settings [24]. Compared with resource-intensive SOTA approaches such as ARF and ODL, the proposed framework substantially reduces computational overhead (97 updates versus 249) while maintaining stable and efficient latency (~ 769 ms per adaptation cycle). Moreover, the entropy-driven drift signal is shown to be more effective than classical techniques such as ADWIN, suggesting a promising direction for low-resource adaptive security systems. Results on complex composite streaming scenarios further indicate that CD-AHAL outperforms static models and representative baselines in terms of rapid accuracy recovery and sustained F1-score (achieving 90.77% in binary and 77.33% in multiclass scenarios), enabled by an oracle-query mechanism under drift.

Future work will focus on two directions to strengthen real-world applicability: (1) operationalizing the oracle by directly integrating real-time threat-analysis services (e.g., VirusTotal, AbuseIPDB) or developing specialized teacher models for automated labeling,

rather than relying on simulated ground truth; and (2) further reducing computational cost via knowledge distillation to enable even more efficient deployment on highly resource-constrained edge devices.

REFERENCES

- [1] M. Musthafa, “Real-time intrusion detection leveraging deep learning: A comparative analysis of CNN, RNN, and transformer architectures,” *International Journal of Advanced Engineering, Management and Science*, vol. 11, no. 5, pp. 64–72, 2025. [Online]. Available: <https://doi.org/10.22161/ijaems.115.8>
- [2] M. Uğurlu and İ. A. Doğru, “A survey on deep learning based intrusion detection system,” *2019 4th International Conference on Computer Science and Engineering (UBMK)*, pp. 223–228, 2019. [Online]. Available: <https://doi.org/10.1109/UBMK.2019.8907206>
- [3] T. T. Nguyen, U. H. Tran, and H. N. Nguyen, “Adversarial attack and defense in AI-powered intrusion detection,” *Journal of Computer Science and Cybernetics*, vol. 41, no. 4, pp. 387–402, 2025. [Online]. Available: <https://doi.org/10.15625/1813-9663/22884>
- [4] M. Kumar, “Adaptive defense mechanisms for improving adversarial robustness of deep learning-based intrusion detection systems,” *2026 International Conference on Electronic Systems and Intelligent Computing (ICESIC)*, pp. 988–993, 2026. [Online]. Available: <https://ieeexplore.ieee.org/document/11496452>
- [5] G. D’Angelo, M. Ficco, and F. Palmieri, “Inter-layer autoencoder for mitigating evasion attacks against network intrusion detection systems,” *Applied Soft Computing*, vol. 197, 2026, article no. 115185. [Online]. Available: <https://doi.org/10.1016/j.asoc.2026.115185>
- [6] N. Gönitz, M. Kloft, K. Rieck, and U. Brefeld, “Active learning for network intrusion detection,” *Proceedings of the 2nd ACM Workshop on Security and Artificial Intelligence*, pp. 47–54, 2009. [Online]. Available: <https://doi.org/10.1145/1654988.1655002>
- [7] B. Jiao *et al.*, “A semi-supervised active learning neural network for data streams with concept drift,” *IEEE Computational Intelligence Magazine*, vol. 20, no. 1, pp. 18–33, 2025. [Online]. Available: <https://doi.org/10.1109/MCI.2024.3486680>
- [8] Y. Gal and Z. Ghahramani, “Dropout as a Bayesian approximation: Representing model uncertainty in deep learning,” *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, vol. 48, pp. 1050–1059, 2016, pMLR. [Online]. Available: <https://proceedings.mlr.press/v48/gal16.pdf>
- [9] M. Sajid *et al.*, “Enhancing intrusion detection: a hybrid machine and deep learning approach,” *Journal of Cloud Computing*, vol. 13, no. 1, 2024, article no. 123. [Online]. Available: <https://doi.org/10.1186/s13677-024-00685-x>
- [10] C. Yin, Y. Zhu, J. Fei, and X. He, “A deep learning approach for intrusion detection using recurrent neural networks,” *IEEE Access*, vol. 5, pp. 21 954–21 961, 2017. [Online]. Available: <https://doi.org/10.1109/ACCESS.2017.2762418>
- [11] S. Cai *et al.*, “CDDA-MD: An efficient malicious traffic detection method based on concept drift detection and adaptation technique,” *Computers & Security*, vol. 148, 2025, article no. 104121. [Online]. Available: <https://doi.org/10.1016/j.cose.2024.104121>
- [12] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: Synthetic minority over-sampling technique,” *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002. [Online]. Available: <https://doi.org/10.1613/jair.953>
- [13] R. S. Miani *et al.*, “A survey of data stream-based intrusion detection systems,” *IEEE Access*, vol. 13, pp. 72 953–72 983, 2025. [Online]. Available: <https://doi.org/10.1109/ACCESS.2025.3561105>
- [14] D. Sahoo, Q. Pham, J. Lu, and S. C. H. Hoi, “Online deep learning: Learning deep neural networks on the fly,” *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 2660–2666, 2018. [Online]. Available: <https://doi.org/10.24963/ijcai.2018/369>

- [15] A. Bifet and R. Gavaldà, “Learning from time-changing data with adaptive windowing,” *Proceedings of the 2007 SIAM International Conference on Data Mining*, pp. 443–448, 2007. [Online]. Available: <https://doi.org/10.1137/1.9781611972771.42>
- [16] G. Andresini *et al.*, “INSOMNIA: Towards concept-drift robustness in network intrusion detection,” *Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security*, pp. 111–122, 2021. [Online]. Available: <https://doi.org/10.1145/3474369.3486864>
- [17] M. A. Al-Garadi *et al.*, “A survey of machine and deep learning methods for internet of things (IoT) security,” *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1646–1685, 2020. [Online]. Available: <https://doi.org/10.1109/COMST.2020.2988293>
- [18] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, “Toward generating a new intrusion detection dataset and intrusion traffic characterization,” *4th International Conference on Information Systems Security and Privacy (ICISSP)*, pp. 108–116, 2018. [Online]. Available: <https://doi.org/10.5220/0006639801080116>
- [19] E. C. P. Neto *et al.*, “CICIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment,” *Sensors*, vol. 23, no. 13, 2023, article no. 5941. [Online]. Available: <https://doi.org/10.3390/s23135941>
- [20] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, “Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy,” *2019 International Carnahan Conference on Security Technology (ICCST)*, pp. 1–8, 2019. [Online]. Available: <https://doi.org/10.1109/CCST.2019.8888419>
- [21] A. H. Lashkari, G. Kaur, and A. Rahali, “DIDarknet: A contemporary approach to detect and characterize the darknet traffic using deep image learning,” *Proceedings of the 2020 10th International Conference on Communication and Network Security*, pp. 1–13, 2020. [Online]. Available: <https://doi.org/10.1145/3442520.3442521>
- [22] M. MontazeriShatoori, L. Davidson, G. Kaur, and A. H. Lashkari, “Detection of DoH tunnels using time-series classification of encrypted traffic,” *Proc. IEEE DASC/PiCom/CBDCoM/CyberSciTech*, pp. 63–70, 2020. [Online]. Available: <https://doi.org/10.1109/DASC-PiCom-CBDCoM-CyberSciTech49142.2020.00026>
- [23] H. M. Gomes *et al.*, “Adaptive random forests for evolving data stream classification,” *Machine Learning*, vol. 106, no. 9, pp. 1469–1495, 2017. [Online]. Available: <https://doi.org/10.1007/s10994-017-5642-8>
- [24] A. D. Nguyen Phuoc, “CD-AHAL source code,” GitHub Repository, 2026. [Online]. Available: https://github.com/NguyenPhuocAnhDung/CD_AHAL

Received on February 17, 2026

Accepted on March 23, 2026