Journal of Basrah Researches (Sciences) **52**(1), 313 (2026)
DOI: https://DOI.org/10.56714/bjrs.52.1.22

A Lightweight Drift-Aware Continual Learning Framework for IoT Malware Detection Across Evolving Sessions

Sattar J. J. Yahya*, Baraa I. Farhan

Department of Computer Science, College of Education for Pure Sciences, Wasit University, Iraq.

ARTICLE INFO

Received 29 March 2026
Revised 28 April 2026
Accepted 07 May 2026
Published 30 June 2026

Keywords:

Iot Malware Detection, Continual Learning, Concept Drift, PSI, Iot-23.

Citation: S. J. J. Yahya, B. I. Farhan ., J. Basrah Res. (Sci.) **52**(1), 313 (2026). DOI:https://DOI.org/10.56714/bjrs.52.1.22

ABSTRACT

IoT malware detection is commonly evaluated under static train–test settings, whereas real-world deployments operate over continuous traffic streams where both benign behavior and attack patterns evolve over time. This paper presents a lightweight drift-aware continual learning framework for IoT malware detection, evaluated on the IoT-23 dataset within a session-based evolution setting. The model is built on a compact multilayer perceptron and examined under six continual learning configurations, including replay, Elastic Weight Consolidation (EWC), Synaptic Intelligence (SI), and their combinations.

A prequential evaluation protocol is adopted, where each incoming session is first evaluated and then incrementally incorporated into the model, rather than relying solely on final test outcomes. To better understand distributional changes, the workflow tracks Population Stability Index (PSI) across consecutive sessions and reports forgetting behavior alongside Accuracy and Macro-F1.

In the re-evaluated results, the NR_EWC configuration achieved the most balanced overall performance, with an average Macro-F1 of 0.7918, an average accuracy of 0.9887, and moderate forgetting (0.0660). In contrast, the NR_NoReg baseline achieved high accuracy (0.9864) but exhibited the highest forgetting (0.2137), indicating that accuracy alone may mask stability limitations under drift conditions.

1. Introduction

Internet-of-Things (IoT) settings are not run as structured and clean datasets. They are long-lived traffic streams where devices behavior is modified over time, routines of use change, benign patterns of communication drift, and the malicious activity can take new forms. This practical fact imposes a sharp disjuncture with significant portion of the literature on the IoT malware-detection due to the fact that models are still evaluated based on a single fixed split and then assumed to perform similarly when deployed [1]- [3].

Simultaneously, new developments have taken two significant turns. Adaptation, replay, and drift-aware learning are discussed in one set of literature, which gives it a certain understanding that forgetting and distribution shift have ceased to be secondary concerns in security analytics [4]- [6].

***Corresponding author email:** Sattaryehya@gmail.com

ISSN: 1817-2695 (Print); 2411-524X (Online)
Online at: https://jou.jobrs.edu.iq

The second group is still advocating models that are smaller, faster, and can be deployed in limited environments, but these advantages are typically tested in a static and not dynamic operating environment.
Thus, the question this paper asks is a bit more practical and focused: can a small continual learning workflow be useful when considering the evaluation of IoT malware detection as a process of evolving sessions rather than a single static split? To achieve this goal, the paper uses the well-known benchmark of IoT-23, which is still one of the most useful references due to the presence of labeled malicious and benign IoT captures of real-world scenarios [1]. The goal is not to propose a new architecture; rather, to propose a combination of a base learner and session evaluation within a single experiment.
The second motivation comes from methodological concerns. In IoT security problems, a high value of accuracy does not guarantee that the minority class is performing well or that it does not forget what it learned during adaptation. Recent research in adaptive IoT security, feature-efficient intrusion analysis, and generalization-aware lightweight detection has reinforced the need to view the results of a classification problem more than what a single value of accuracy or another single value of a different metric permits [7]-[9]. As such, the current paper examines the results from more than a single perspective. In addition to Accuracy and Macro-F1, it also examines pre-update degradation, forgetting tendencies, and Population Stability Index over session transitions.
The main contributions of this paper can be summarized as follows:

- A compact drift-aware continual-learning workflow for IoT malware detection is presented using a lightweight MLP, prequential evaluation, session-wise updating, and PSI-based drift interpretation in one unified pipeline.
- Six continual-learning configurations are compared under the same ordered IoT-23 session protocol, enabling a fair analysis of replay, EWC, SI, and their hybrid combinations.
- The paper provides a balanced empirical interpretation that separates raw accuracy from Macro-F1, pre-update difficulty, and forgetting, showing why the best continual configuration is not necessarily the one with the highest average accuracy.
- A methodological presentation is supplied through a professional workflow diagram, a formal algorithm, and mathematical expressions for the key optimization and evaluation components.

The remainder of this paper is organized as follows. Section 2 reviews the most relevant studies on lightweight and adaptive IoT malware detection. Section 3 presents the proposed framework, the mathematical formulation, the evaluation algorithm, and the main experimental settings. Section 4 reports and discusses the empirical results, including the overall comparison, session-wise analysis, contextual comparison with prior studies, and latency profiling. Finally, Section 5 concludes the paper and outlines realistic directions for future work

2. Related Work

The current research on IoT malware detection can be classified into three directions that are quite similar. The former direction deals with lightweight detection. Research in this field attempts to make the model small enough to fit in limited environments but with predictive power useful to the researcher. It is important since most IoT deployments are unable to run heavy training or inference pipelines over extended durations [3], [6], [10], [11].
The second direction is that of adaptation where the data no longer act in a stable manner. Furthermore, more recent works have considered the replay, constant update, drift resiliency, and benchmark design on a dynamic traffic environment. This tendency is symptomatic of a practical issue: a model that can do well on a single fixed split can fail at a later moment when the device behaviour, attack mix or timing of traffic changes [4] [5], [7], [8], [12], [13].
The third direction is more extensive methodological support of the IoT security research such as efficient feature selection, robust model design, and massive comparative assessment. These investigations drive the discipline to less idealistic and practicable experimentation, however the majority remain to focus on fixed or semi-fixed appraisal instead of ordered session strict test-before-update learning [9], [14], [15], [16], [17], [18]. It is in this context that the current paper follows a narrower scope. It does not attempt to surpass all the static outputs in the literature. Rather, it posits whether a small continuous learning process can be trusted to be reliable with IoT malware detection

being tested with session evolution, as well as whether the same behavior is more understandable when drift and forgetting are reported simultaneously.

As can be seen in Table 1, more and more studies focus on efficiency, adaptive behavior, or benchmarking quality, although only a small portion of them considers more than one of these necessities simultaneously. Precisely, lightweight IoT malware papers tend to stick at rest, whereas drift-oriented or benchmark-oriented papers tend to terminate before actual session-wise prequential updating. The current work is placed here: a small flow-based malware analysis makes the model small enough, applies it step-by-step, and interprets drift directly in lieu of post-update accuracy as a good enough indicator.

Table 1. Comparative summary of representative recent studies related to lightweight and adaptive IoT malware detection.

Study (Author/ Year)	Algorithm/ Method	Strength	Limitation	Best Use Case	Accuracy	F1-score	Computational Cost	Continual Learning Support	Context-Aware Capability
Farfoura et al. (2025)	MBMD + lightweight ML	Designed for limited computing burden; strong malware detection	Evaluated in a largely static setting; no session-wise continual update	Lightweight batch malware detection	NR	>99%	Low	No	No
Al-Fawa'reh et al. (2024)	Deep reinforcement learning	Explicitly targets drift and dynamic botnet behavior	Heavier than compact tabular pipelines; not prequential on IoT-23 sessions	Adaptive IoT botnet detection	≈99.9 %	NR	High	Partial	No
Patel et al. (2025)	Federated RF + drift detectors	Addresses local concept drift in distributed malware learning	Federated Android setting differs from centralized IoT traffic streams	Privacy-preserving distributed malware detection	94.7–96.8%	NR	Moderate	Yes	Partial
Kouassi et al. (2025)	Top-K feature selection + RF	Good balance between performance and efficiency; 35% time reduction	Not a continual-learning design; focuses on static IDS evaluation	Feature-efficient IoT IDS	91.7%	93.0%	Low	No	No
Wakili et al. (2025)	Hybrid feature selection + explainable ensemble	High multiclass accuracy with explainability	Macro-F1 remains lower than accuracy; no forgetting control	Resilient multi-class IoT intrusion detection	99.16 %	73.79 %	Moderate –High	No	No
Lyon et al. (2026)	RF, LightGBM, LR, MLP benchmark	Directly studies IoT-23 under temporal robustness and data scarcity	Benchmarking-oriented; does not combine replay, EWC, SI, and PSI together	Temporal robustness benchmarking on IoT-23	99.81 %	97.71 %	Moderate	No	No
Sait et al. (2026)	Hybrid malware detection model	Strong IoT-23 performance with explicit inference-time reporting	Focuses on classification performance rather than continual adaptation	Accurate low-latency IoT malware classification	98.23 %	96.71 %	Low–Moderate	No	No

3. Proposed Framework and Experimental Design

The workflow we propose for this task is depicted in Fig. 1. The preprocessed and standardized, but otherwise ordered, IoT-23 sessions are processed one by one. Before each session's update, the current model's performance is evaluated, providing the prequential notion of the degree to which the learned representation has deteriorated due to the drift. Then, the appropriate policy for continual learning is applied, and the metrics are recorded after the update. In this study, the learner is intentionally designed to be lightweight, specifically a shallow feed-forward multilayer perceptron (MLP), to maintain analytical clarity and avoid unnecessary architectural complexity. This design choice allows the study to focus on the behavior of continual learning under session evolution without introducing confounding architectural factors. For an input flow vector $\boldsymbol{x} \in \boldsymbol{R^d}$, the lightweight classifier produces a hidden representation h and an output probability $\hat{y}$ for the malware class. The basic forward mapping can be summarized as:

$$h = ReLU(W^1 x + b^1) \quad \ldots(1)$$

$$\hat{y} = \sigma(W^2 h + b^2) \quad \ldots(2)$$

The optimization objective changes according to the selected continual-learning configuration. The standard classification loss is binary cross-entropy:

$$L_{cls} = -\frac{1}{N} \Sigma_{\{n=1\}}^{\{N\}} [\, y_n \, log(\hat{y}_n) + (1 - y_n) log(1 - \hat{y}_n)] \quad \ldots(3)$$

When replay is enabled, the effective training loss becomes the sum of the current-session loss and a replay loss over stored samples:

$$L_{replay_{total}} = L_{cls}(S_t) + \lambda_r \, L_{cls}(B_t) \quad \ldots(4)$$

EWC constrains important parameters from drifting too far from their previously learned values by using a Fisher-weighted quadratic penalty [19]:

$$L_{EWC} = L_{cls} + \left(\frac{\lambda_e}{2}\right) \Sigma_i \, F_i \, (\theta_i - \theta_i *)^2 \quad \ldots(5)$$

SI plays a similar stabilizing role, but its importance weights are accumulated from the contribution of each parameter during training [20]:

$$L_{SI} = L_{cls} + \lambda_s \, \Sigma_i \, \Omega_i \, (\theta_i - \theta_i *)^2 \quad \ldots(6)$$

In the hybrid configuration used in this paper, the objective can be written compactly as:

$$L_{total} = L_{cls}(S_t) + \lambda_r \, L_{cls}(B_t) + \left(\frac{\lambda_e}{2}\right) \Sigma_i \, F_i \, (\theta_i - \theta_i *)^2 + \lambda_s \, \Sigma_i \, \Omega_i \, (\theta_i - \theta_i *)^2 \quad \ldots(7)$$

The evaluation metrics follow standard definitions. Accuracy and Macro-F1 are used as the main post-update measures:

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \quad \ldots(8)$$

$$Precision = \frac{TP}{(TP + FP)} \quad \ldots(9)$$

$$Recall = \frac{TP}{(TP + FN)} \quad \ldots(10)$$

$$Macro_{F1} = \frac{1}{C}\sum_{c=1}^{C}\frac{2 \cdot Precision_c \cdot Recall_c}{(Precision_c + Recall_c)} \quad \ldots(11)$$

Forgetting is used to summarize how much earlier knowledge deteriorates after later updates:

$$F_t = \left(\frac{1}{(t-1)}\right)\sum_{k=1}^{t-1}[max_{1<t}\, a_{k,l} - a_{k,t}] \quad \ldots(12)$$

To interpret distribution change between two consecutive sessions, the study uses Population Stability Index (PSI):

$$PSI = \Sigma_j(p_j - q_j)ln\left(\frac{p_j}{q_j}\right) \quad \ldots(13)$$

where p_j and q_j are the proportions of feature value which occur in bin j in the reference and current session, respectively. Increase in PSI values show greater distributional change. PSI is utilized in present paper as an interpretation signal instead of an automatic update trigger.

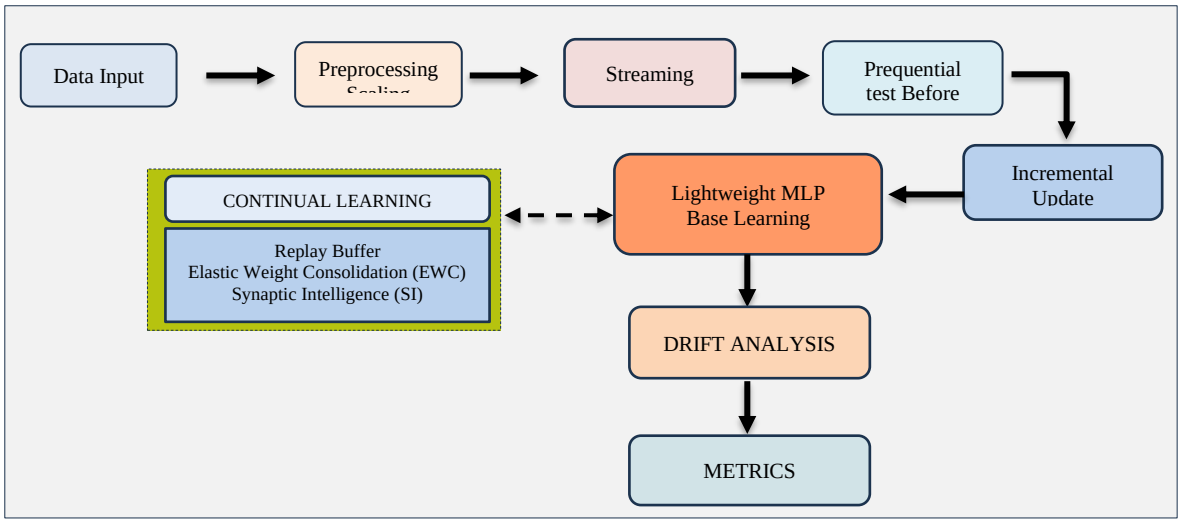

Fig 1. Proposed workflow for drift-aware continual IoT malware detection under session evolution.

Algorithm 1 formalizes the operational sequence used in the experiment.

Algorithm 1. Drift-aware continual evaluation procedure
Input: ordered sessions S = {S1, S2, ..., ST}, configuration c, lightweight MLP f
Output: session-wise pre-update metrics, post-update metrics, forgetting, PSI
1: Train f on the initial session S1 using configuration c.
2: Record post-update metrics for S1 as the starting reference.
3: For each next session St, t = 2 ... T do
4: Compute PSI between St-1 and St after preprocessing.
5: Evaluate the current f on St before update and store prequential metrics.
6: Update f with St using the selected policy in c.
7: If replay is enabled, mix current data with buffer samples.
8: If EWC is enabled, add the Fisher-weighted stability penalty.
9: If SI is enabled, add the path-integral stability penalty.
10: Evaluate the updated f on St and store post-update metrics.
11: Recompute average forgetting from retained past-session performance.
12: End for.
13: Rank configurations using Macro-F1, stability, and forgetting.

Before presenting the measured results, Table 2 summarizes the main experimental settings used in the study.

Table 2. Experimental settings used in the IoT-23 drift-aware evaluation

Item	Setting
Dataset	IoT-23
Evaluation unit	Ordered flow sessions
Base learner	Compact feed-forward MLP
Continual configurations	NR_NoReg, NR_EWC, NR_SI, R_EWC, R_SI, R_EWC_SI
Evaluation protocol	Prequential test before each incremental update
Drift indicator	Population Stability Index (PSI) across consecutive sessions
Reported metrics	Accuracy, Macro-F1, Macro-Precision, Macro-Recall, average forgetting
Primary interpretation goal	Balance adaptation quality, stability, and forgetting under session evolution

Note: NR = No Replay; R = Replay Buffer; EWC = Elastic Weight Consolidation; SI = Synaptic Intelligence.

For creating realistic conditions for the concept drift simulation, IoT-23 was first ordered in terms of time and then divided into five different sessions. This is done to create different traffic segments where attacks are distributed differently and thus test the adaptability of models with evolving data streams. There are over six million flow entries within the dataset, with different attacks being distributed differently within each session.

In order to achieve repeatability and methodology in the experiments, we had to choose the features that we would use in our input from the original IoT-23 flow-based features. These include non-relevant identifiers like unique connection ID (uid), IP address, timestamp, and label, which would have resulted in leakage of information about the class. The end result was a total of 16 features (selected from IoT-23 flow attributes; see Table 5) including statistical flow attributes, packets level features, and categorical features like protocol, service, and connection state.

Table 3. Selected input features used in the proposed model

Feature	Type	Description
Duration	Numeric	Connection duration
orig_bytes	Numeric	Bytes sent from source
resp_bytes	Numeric	Bytes sent from destination
orig_pkts	Numeric	Packets from source
resp_pkts	Numeric	Packets from destination
orig_ip_bytes	Numeric	Total IP bytes from source
resp_ip_bytes	Numeric	Total IP bytes from destination
Proto	Categorical (encoded)	Transport protocol (TCP/UDP/ICMP)
Service	Categorical (encoded)	Application service type
conn_state	Categorical (encoded)	Connection state
local_orig	Binary	Source is local
local_resp	Binary	Destination is local
missed_bytes	Numeric	Missing bytes count
History	Categorical (encoded)	Connection history flags
orig_p	Numeric	Source port
resp_p	Numeric	Destination port

In order to improve reproducibility, it is possible to explicitly describe the network architecture adopted in the experiments. Table 4 describes the practical configuration of the lightweight MLP adopted for the session-wise continual learning evaluation.

Table 4. Configuration of the lightweight MLP used in the IoT-23 experiments.

Component	**Specification**
Model Name	LightweightContinualMLP
Input Features	16
Hidden Layer 1	128 neurons + ReLU + Dropout (0.4)
Hidden Layer 2	64 neurons + ReLU + Dropout (0.3)
Output Layer	2 neurons
Optimizer	Adam
Learning Rate	0.001
Weight Decay	0.0001
Loss Function	FocalLoss (gamma = 1.0)
Batch Size	5000
Replay Buffer Settings	replay_per_class = 800, replay_sample = 2000
EWC Update Settings	ewc_update_every = 20, ewc_fisher_samples = 1024

4. Results and Discussion

The overall comparison is first summarized in Table 4. This is more informative than the individual scores because it combines post-update quality, pre-update difficulty, and forgetting in one place. The most interesting finding is that the best configuration is not the one with the lowest forgetting, but the one with the most balanced predictive performance.

Table 4. Overall performance comparison across the six continual-learning configurations

Configuration	**Avg. Accuracy**	**Avg. Macro-F1**	**Avg. Preq. Macro-F1**	**Final Forgetting**
NR_EWC	0.9887	0.7918	0.4198	0.0660
R_EWC	0.9747	0.7361	0.4323	0.0936
NR_NoReg	0.9864	0.7229	0.4174	0.2137
NR_SI	0.9781	0.6795	0.3799	0.1706
R_SI	0.7500	0.5806	0.4312	0.0715
R_EWC_SI	0.7796	0.5688	0.4372	0.0066

In order to have a concise visual view of the results, Fig. 2 illustrates the average results of the six different settings' performances. As shown in Fig. 2, the highest balance was obtained by the NR_EWC method in the re-execution of the settings, while different trade-offs were observed for the other settings in terms of adaptation quality, stability, and retention.

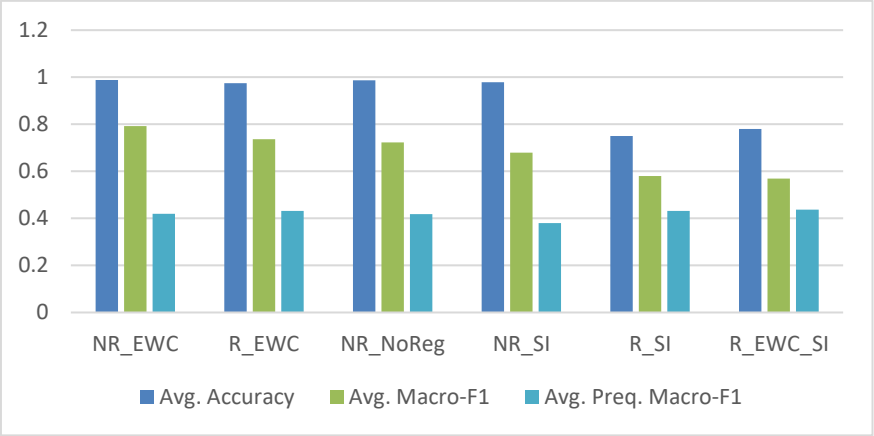

Fig 2. Average performance metrics across IoT-23 sessions for the evaluated continual-learning configurations.

The average comparison also fails to capture the extent to which the stream varied from one session to the next. For this reason, Table 5 also reports the Macro-F1 for each session post-update. This is because drift is rarely uniform in practice, and a method may look stable on average but struggle at a certain point.

Table 5. Session-wise post-update Macro-F1 across IoT-23 sessions.

Configuration	Session 01	Session 02	Session 03	Session 04
NR_EWC	0.6733	0.9148	0.8986	0.6805
NR_NoReg	0.8324	0.8886	0.4948	0.6759
NR_SI	0.5517	0.8669	0.6133	0.6863
R_EWC	0.8919	0.5802	0.6166	0.8558
R_EWC_SI	0.4169	0.4142	0.5817	0.8623
R_SI	0.8772	0.8662	0.4882	0.0911

Tables 4 and 5 indicates some obvious conclusions. To begin with, the maximum average accuracy in itself was not a sufficient criterion to assess the quality of continual-learning during evolution of the session. However, in spite of the fact that NR_NoReg was quite robust in the average accuracy, its ultimate forgetting was the most aggressive of all the tested settings, a fact that implies a worse retention of previously learned behavior over a period of time.

Second, the re-calculated findings suggest that NR-EWC delivered the best balance on the whole in this research. It recorded the best average Macro-F1, 0.7918 and the best average accuracy, 0.9887 with a moderate forgetting, 0.0660. This trend indicates that EWC-based stabilization allowed to conserve valuable knowledge in the session-to-session without the greater instability exhibited by certain replay-based combinations.

Third, the session view reveals that the stream did not have a consistent level of difficulty. Session 03 was difficult on a few settings, and Session 04 revealed evident weaknesses on certain settings, such as RSI. This difference affirms the fact that development of sessions cannot be regarded as a gradual or foreseeable phenomenon. Rather, various continual-learning policies reacted differently to various transitions.

To provide a clearer view of session-level behavior, Fig. 3 illustrates the post-update Macro-F1 trends across the six evaluated continual-learning configurations. The figure highlights how performance changed from one session to another and makes the variability of adaptation quality easier to observe than the table alone.

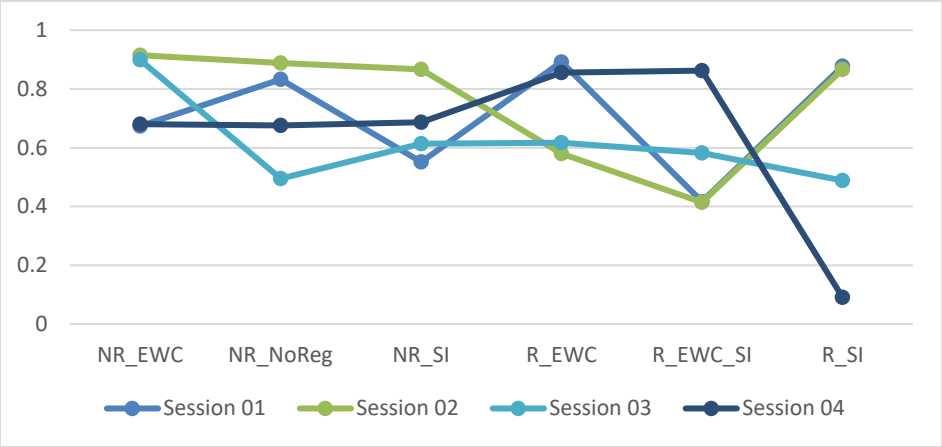

Fig 3. Session-wise post-update Macro-F1 trends across the six continual-learning configurations on IoT-23.

The PSI values help explain this behavior. The average transition values were about 0.8252, 0.7823, and 0.8755 across sessions 01-02, 02-03, and 03-04, respectively. These values indicate substantial movement in the data distribution, which helps explain why pre-update scores remained modest and why test-before-update evaluation remained necessary throughout the stream.

From a practical perspective, the results support a more careful conclusion than the one often implied by static benchmarking. A lightweight model may still be useful in changing conditions of the IoT traffic, yet the credibility of the model is determined by the quality of adaptation, the balance of the classes, and forgetting. In this re-implementation, the EWC-based set-up provided the most persuasive balance between predictive power and stored information.

It is helpful to juxtapose the main finding of this study with representative precedent work before coming to a conclusion. Table 6 is a contextual and not absolute comparison due to the difference in data sets, description of tasks, and protocols used in the cited studies. Nevertheless, it can be used to explain the value added by the current work: the results obtained were reported in a more challenging session-wise constant environment, but many of the past studies had higher headline scores in a static assessment.

Table 6. Contextual comparison between representative prior studies and the best result of this study

Study (Author/Year)	Dataset / Setting	Accuracy	F1-score	Evaluation Mode	Computational Cost	Continual Learning / Drift Handling
Farfoura et al. (2025) [6]	IoT malware classification	NR	>99%	Static split	Low	No continual update; no explicit drift analysis
Chen et al. (2025) [13]	Lightweight host-level IoT malware detection	NR	High	Static evaluation	Low	No continual update; limited drift discussion
Lyon et al. (2026) [16]	IoT-23 with scarcity and temporal drift	NR	0.9982 (bin.) / 0.8531 (multi.)	Temporal benchmark	Moderate	Drift-aware benchmark; not test-before-update continual learning
Sait et al. (2026) [17]	IoT malware detection	99.35%	98.42%	Static split	Moderate	No continual update; no session-wise drift protocol
Maghanaki et al. (2026) [18]	Cross-family IoT malware evaluation	NR	Strong multiclass results	Comparative static evaluation	Moderate-High	Broad comparison; no continual adaptation
This study	IoT-23, session-wise continual detection	0.9887	0.7918	Prequential test-before-update	Low to Moderate	EWC-based continual updating with forgetting and PSI

Besides the evaluation of the accuracy of predictions, it is critical to analyze the computational characteristics of the selected configurations. Specifically, a profiling run was performed within the same experiment pipeline to determine how long it takes to train the models and make predictions. The metrics related to computation are not employed to change the rankings based on prediction accuracy; instead, they help gain insight into practical deployment aspects.
As may be observed from Table 7, the proposed MLP was observed to remain lightweight from an inference perspective across all configurations. This indicates that, even when replay or regularization was used and had a significant effect on update cost, the average inference time per sample was extremely small, further supporting the suitability of the workflow.

Table 7. Latency analysis across the evaluated continual-learning configurations (profiling run).

Configuration	Training Time / Session (s)	Preq. Inference (s)	Test Inference (s)	Session Total (s)	Preq. ms/sample	Test ms/sample
NR_SI	6.8137	0.0290	0.0286	7.6544	0.000581	0.000572
NR_NoReg	7.8582	0.0354	0.0468	8.7504	0.000708	0.000936
R_SI	23.9024	0.0721	0.0687	25.5158	0.001442	0.001374
R_EWC	30.6938	0.0469	0.0437	31.7928	0.000938	0.000873
R_EWC_SI	33.2813	0.0494	0.0433	34.4377	0.000988	0.000867
NR_EWC	42.7737	0.0590	0.0538	44.1706	0.001180	0.001077

The results on latency have also demonstrated the expected computational trade-off, where the best session-level execution was observed for NR_SI and NR_NoReg, and NR_EWC was observed to have the largest average update cost due to the added consolidation step. However, even this was observed to be extremely low, further supporting the suitability of the workflow for edge-oriented IoT settings, where inference efficiency is of primary importance rather than on-device retraining.
To give a clearer deployment-oriented view, Fig. 3 compares the average inference latency per sample across the evaluated configurations. The figure shows that inference remained very small in all settings, despite differences in update strategy.

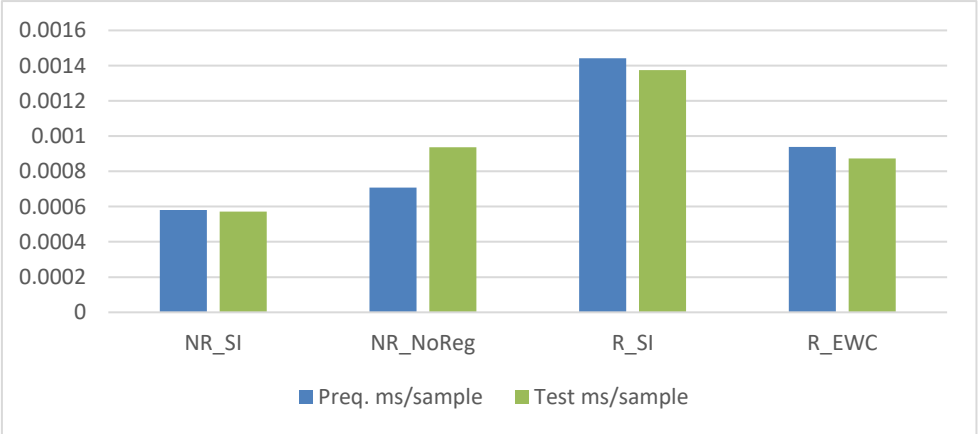

Fig 3. Average inference latency per sample across the evaluated continual-learning configurations.

5. Conclusion and Future Work

In this paper, the authors have introduced a lightweight drift-conscious framework of continual learning to detect IoT malware and tested it on IoT-23 when changing sessions. The re-performed results demonstrated that NR_EWC offered the best overall performance balance and a Macro-F1 of 0.7918, a mean accuracy of 0.9887, and average forgetting than the other settings. The findings also indicated that high-accuracy was not sufficient to ensure reliable continuity behavior because designs with great average accuracy may be characterized by poorer between-session retention.

The research also revealed that a single split is not as realistic as session-wise evaluation of the behavior of the model. The paper introduces the combination of prequential testing, forgetting analysis, and PSI-based drift interpretation as the means of how lightweight IoT malware detection may be evaluated more fairly under changing traffic conditions.

The future work must be realistic and deployment based. One, the identical protocol may be ran on another large dataset of IoT to check whether the same ranking will be consistent with another traffic profile. Second, PSI may be used in conjunction with explicit update-trigger policy in such a way that adaptation is only done when the observed shift is larger than a reasonable threshold. Third, the framework may be expanded to binary malware detection to what is known as malware-family or attack-type classification. Lastly, the workflow can be deployed in an edge-gateway prototype in a manner that enables measuring memory utilization, update cost, and latency to be under long-standing streams of traffic.

References

[1] S. Garcia, A. Parmisano, and M. J. Erquiaga, "IoT-23: A labeled dataset with malicious and benign IoT network traffic," Zenodo, 2020, DOI: 10.5281/zenodo.4743746.

[2] E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, and A. A. Ghorbani, "CICIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment," Sensors, vol. 23, no. 13, p. 5941, 2023, DOI: 10.3390/s23135941.

[3] J. Ramamoorthy, K. Gupta, R. C. Kafle, N. K. Shashidhar, and C. Varol, "A novel static analysis approach using system calls for Linux IoT malware detection," Electronics, vol. 13, no. 15, p. 2906, 2024, DOI: 10.3390/electronics13152906.

[4] M. Chin and R. Corizzo, "Continual semi-supervised malware detection," Machine Learning and Knowledge Extraction, vol. 6, no. 4, pp. 2829–2854, 2024, DOI: 10.3390/make6040135.

[5] M. Al-Fawa'reh, J. Abu-Khalaf, P. Szewczyk, and J. J. Kang, "MalBoT-DRL: Malware botnet detection using deep reinforcement learning in IoT networks," IEEE Internet of Things Journal, vol. 11, no. 6, pp. 9610–9629, 2024, DOI: 10.1109/JIOT.2023.3324053.

[6] M. E. Farfoura et al., "A novel lightweight machine learning framework for IoT malware classification based on matrix block mean downsampling," Ain Shams Engineering Journal, vol. 16, no. 1, p. 103205, 2025, DOI: 10.1016/j.asej.2024.103205.

[7] N. A. H. Fathima et al., "Adaptive memory replay for network intrusion detection: Tackling data drift and catastrophic forgetting," Computer Networks, 2025, DOI: 10.1016/j.comnet.2025.111712.

[8] C. Guo, X. Li, J. Cheng, S. Yang, and H. Gong, "Continual learning for intrusion detection under evolving network threats," Future Internet, vol. 17, no. 10, p. 456, 2025, DOI: 10.3390/fi17100456.

[9] I. Mutambik et al., "AI-driven cybersecurity in IoT: Adaptive malware detection and lightweight encryption via TRIM-SEC framework," Sensors, vol. 25, no. 22, p. 7072, 2025, DOI: 10.3390/s25227072.

[10] Z. Chen et al., "HoleMal: A lightweight IoT malware detection framework based on efficient host-level traffic processing," Computers & Security, vol. 151, p. 104206, 2025, DOI: 10.1016/j.cose.2025.104206.

[11] M. R. A. Khan et al., "Lightweight quantized XGBoost for botnet detection in resource-constrained IoT networks," IoT, vol. 6, no. 4, p. 70, 2025, DOI: 10.3390/iot6040070.

[12] A. Patel et al., "FL-MalDrift: A federated learning framework for malware detection under local concept drift," Scientific Reports, vol. 16, p. 1821, 2026, DOI: 10.1038/s41598-025-31592-z.

[13] J. Lyon, K. Goyal, and C. Doerr, "Benchmarking machine learning models for IoT malware detection under data scarcity and drift," arXiv preprint arXiv:2601.18736, 2026.

[14] B. M. Kouassi et al., "Top-K feature selection for IoT intrusion detection: Contributions of XGBoost, LightGBM, and random forest," Future Internet, vol. 17, no. 11, p. 529, 2025, DOI: 10.3390/fi17110529.

[15] Y. M. Yang et al., "A lightweight training approach for MITM detection in IoT networks: Time-window selection and generalization," Applied Sciences, vol. 15, no. 22, p. 12147, 2025, DOI: 10.3390/app152212147.

[16] A. Wakili and S. Bakkali, "A resilient IoT intrusion detection system using hybrid feature selection and explainable ensemble learning," Results in Engineering, vol. 30, p. 107392, 2025, DOI: 10.1016/j.rineng.2025.107392.
[17] A. R. W. Sait et al., "Hybrid malware detection model for Internet of Things environment," Journal of Network and Computer Applications, vol. 242, p. 104355, 2026, DOI: 10.1016/j.jnca.2026.104355.
[18] M. Maghanaki et al., "Systematic evaluation of machine learning and deep learning models for IoT malware detection across multiple attack types," Sensors, vol. 26, no. 6, p. 1750, 2026, DOI: 10.3390/s26061750.
[19] J. Kirkpatrick et al., "Overcoming catastrophic forgetting in neural networks," Proc. Natl. Acad. Sci., vol. 114, no. 13, pp. 3521–3526, 2017, DOI: 10.1073/pnas.1611835114.
[20] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in Proc. ICML, vol. 70, pp. 3987–3995, 2017.

Journal of Basrah Researches (Sciences) **52**(1), 313 (2026)
DOI: https://DOI.org/10.56714/bjrs.52.1.22

إطار خفيف الوزن للتعلّم المستمر الواعي بالانجراف لكشف البرمجيات الخبيثة في بيئات إنترنت الأشياء عبر الجلسات المتغيرة

ستار جبار جواد يحيى*، براء اسماعيل فرحان

قسم علوم الحاسوب، كلية التربية للعلوم الصرفة، جامعة واسط، العراق.

معلومات البحث

الاستلام 29 اذار 2026
المراجعة 28 نيسان 2026
القبول 07 أيار 2026
النشر 30 حزيران 2026

الكلمات المفتاحية

اكتشاف البرمجيات الخبيثة لإنترنت الأشياء، التعلم المستمر، انحراف المفاهيم، PSI، Iot-23.

الملخص

يُقيَّم كشف البرمجيات الخبيثة في إنترنت الأشياء عادةً ضمن إعدادات تدريب–اختبار ثابتة، في حين أن الأنظمة الواقعية تعمل على تدفقات مرورية مستمرة حيث يتغير كلٌّ من السلوك الطبيعي وأنماط الهجوم بمرور الوقت. تقدّم هذه الورقة إطارًا خفيف الوزن للتعلّم المستمر الواعي بالانجراف لكشف البرمجيات الخبيثة في إنترنت الأشياء، وتم تقييمه باستخدام مجموعة بيانات IoT-23 ضمن سيناريو قائم على تطور الجلسات. يعتمد النموذج على شبكة عصبية متعددة الطبقات مدمجة، ويُختبر ضمن ستة إعدادات للتعلّم المستمر، تشمل replay و Elastic Weight Consolidation (EWC) و Synaptic Intelligence (SI) وتوليفاتها. تم اعتماد بروتوكول تقييم متتابع، حيث يتم تقييم كل جلسة واردة أولًا ثم دمجها تدريجيًا في النموذج، بدلًا من الاعتماد على نتائج الاختبار النهائية فقط. ومن أجل فهم أفضل للتغيرات في التوزيع، يتتبع إطار العمل مؤشر استقرار التوزيع (PSI) عبر الجلسات المتعاقبة، ويعرض سلوك النسيان إلى جانب الدقة و Macro-F1. في النتائج المعاد تقييمها، حقق إعداد NR_EWC أفضل توازن إجمالي في الأداء، بمتوسط Macro-F1 بلغ 0.7918، ومتوسط دقة 0.9887، ومستوى نسيان متوسط (0.0660). في المقابل، حقق خط الأساس NR_NoReg دقة عالية (0.9864)، لكنه أظهر أعلى مستوى من النسيان (0.2137)، مما يشير إلى أن الاعتماد على الدقة وحدها قد يخفي قيود الاستقرار في ظل الانجراف.

Citation: S. J. J. Yahya, B. I. Farhan ., J. Basrah Res. (Sci.) **52**(1), 313 (2026). DOI:https://DOI.org/10.56714/bjrs.52.1.22

***Corresponding author email:** Sattaryehya@gmail.com

ISSN: 1817-2695 (Print); 2411-524X (Online)
Online at: https://jou.jobrs.edu.iq