electronics

MDPI

Article

Federated Continual Learning for Encrypted Traffic Classification at the Network Edge Under Asynchronous Concept Drift

Abdulrahman K. Alnaim [1,*] and Ahmed M. Alwakeel [2]

[1] Department of Management Information Systems, College of Business Administration, King Faisal University, Alahsa 31982, Saudi Arabia
[2] Faculty of Computers & Information Technology, University of Tabuk, Tabuk 71491, Saudi Arabia; aalwakeel@ut.edu.sa
* Correspondence: aalnaim@kfu.edu.sa

Abstract

Concept drift can degrade encrypted-traffic classifiers deployed at the network edge as applications, protocols, and usage patterns evolve. This paper formulates federated continual learning under asynchronous real- and virtual drift and proposes DriftGuard, a framework combining a two-level per-node detector, selective adapter-based adaptation with Fisher importance masking and class-balanced replay, and drift-aware server aggregation. Level 1 detects distributional changes in learned representations, while Level 2 monitors supervised prediction errors to provide evidence consistent with decision-relevant drift before selective adaptation is activated. We further derive a convergence bound under stated assumptions that explicitly incorporates environmental variation, detection delay, and false alarms. DriftGuard is evaluated in a controlled simulation using reproducible synthetic traffic-like features under sudden, gradual, virtual-only, and recurring drift. Across five independent runs, results are reported with standard deviations, 95% confidence intervals, and paired statistical comparisons. DriftGuard maintains competitive classification accuracy while limiting forgetting of stable classes, with its clearest advantage observed under gradual asynchronous drift. Results also show that immediate adaptation using ground-truth drift states does not necessarily improve performance under the evaluated adaptation policy. The findings provide controlled methodological validation rather than evidence of production-scale deployment performance.

Keywords: federated learning; continual learning; concept drift; encrypted traffic classification; catastrophic forgetting; edge computing; non-IID data; drift detection

Academic Editors: Aryya Gangopadhyay, Pretom Roy Ovi and Sergei Chuprov

Received: 30 July 2026
Revised: 1 September 2026
Accepted: 1 September 2026
Published: 4 September 2026

1. Introduction

Encrypting is no longer a characteristic of sensitive traffic; rather, the Internet is encrypted by default [1]. The vast majority of web, mobile and machine-to-machine communications are now using TLS, QUIC or application-layer encrypted channels, making the techniques used to inspect the payloads of network traffic, which used to underlie traffic engineering, QoS enforcement, and security monitoring, prior to these becoming legacy, largely rendered moot. Network operators are, in response, shifting the burden of classification to machine learning from reading payloads; instead, modern classifiers will predict the application/service and/or threat class of a flow based on side channel observables like packet sizes, inter-arrival times, handshake metadata and burst structure. Pre-trained models based on these observables have been very successful, achieving F1 scores of >97%

on the standard encrypted-traffic benchmarks [2] with transformer representations. These classifiers seem to be a solved problem; when measured on a split of the dataset they were trained on, they held out.

It is a different story when it comes to deployment. The statistical relationship between traffic observables and class labels changes over time: applications release protocol updates, content delivery networks (CDNs) move network resources, TLS and QUIC versions change, user behavior shifts, and new applications emerge and old ones are phased out. The ability of state-of-the-art deep encrypted-traffic classifiers to retain accuracy when deployed is demonstrated to decay significantly over time—even within months of deployment, even without modifications to the model—and rate drop can be so high that even after two years between the training and inference, flows of well-represented classes are no longer well-ranked in the top three predictions [3]. This phenomenon can be considered as an example of concept drift: the changes in the joint distribution of inputs and labels over time that the learning system observes [4]. One of the taxonomies for drift is real-drift, where the conditional relationship between observables and labels changes, the decision boundary needs to change, and in virtual drift, only the input distribution changes, while the boundary remains valid [5]. The difference is crucial in practice; if drift really is happening, it is time to adapt the model, and if the drift is virtual, then computation is wasted, and a model that is correct may be degraded. However, production traffic classifiers do not generally make this split. What is predominant and typical in the industry is periodic wholesale re-training, which is indiscriminate in just this way: it is expensive in labeling and calculation. It is reactive by its very nature: it assumes the model remains the same for the entire time between the onset of drift and the next period of re-training.

The problem becomes even more challenging in the environments where these classifiers are increasingly deployed. The traditional centralized approach to traffic classification is shifting to the network edge, where it needs to be able to operate near the access networks to enable the enforcement of policies with short delays and the local response to threats. Edge deployment of the data distributes the data across the edge nodes [6]. Each edge node sees only its own traffic mix, which is dependent on the local population of users, and the most privacy-sensitive data that the operator owns is its traffic mix. Therefore, it will not be attractive from a contractual or a data-protection regulation point of view to centralize raw flows or features for re-training. Federated learning is the standard approach to training a shared model via the exchange of parameter updates, not the data, and there is an extensive body of work on statistical and systems issues that arises with federated learning [7]. However, as typically defined federated learning is an optimization with a fixed objective and fixed distribution of data, it provides no information on where the data might be located, and it does not provide any information about what to do if the data change under the model.

The study of the temporal dimension has gained much interest from the machine learning community, with its main challenge being catastrophic forgetting, which refers to the loss of memory of a network when it is presented with new data that makes it lose the ability to correctly recognize old data it no longer encounters [8]. Three main families of methods are prevalent: regularization-based approaches constrain parameters that are important to previously learned tasks; replay-based approaches revisit stored or generated examples; and parameter-isolation approaches allocate separate parameters to different tasks [9]. More recently, federated continual learning has surfaced, where the question is: how, as distributed clients receive streams of evolving data, can they learn from the local stream and share the knowledge with a server? [10] It is this very intersection that is where edge traffic classification is born, and data heterogeneity in federated systems is explicitly

identified as a very open problem in the literature [11], with temporally changing traffic distributions of clients being a key one.

There are still existing federated continual learning (FCL) methods which assume a traffic setting that contradicts these assumptions. Most of them are the offshoots of task-incremental continual learning, which assumes that the changes happen as identifiable, incremental tasks—the boundaries of which are known—in contrast to the operational traffic that changes but is unannounced, may be gradual or abrupt, and must be detected, not declared. Most implicitly assume, or assume by experimental design, that the same task sequences are followed on a similar schedule by most of the clients. In an edge deployment, drift is asynchronous: application updates are deployed days before others are deployed at another regional network; some user populations are first to adopt a new service, whereas others delay that adoption; and at any given time some edge nodes are in drift while others are stable. This creates a destructive two-way effect under federated averaging. Changes in the drifted nodes, based on a shifted distribution, influence a common model for the stable majority, while at the same time, the aggregated force of the stable majority tends to dampen just what the drifted minority requires. Finally, there is no previously available framework that would distinguish between real-drift and virtual drift at the point of detection, and therefore adaptation occurs on true and false detections alike and forgetting could be at risk even if the decision boundary is valid.

In this paper, we fill this gap by introducing a federated continual learning framework for edge-based encrypted traffic classification with concept drift in an asynchronous and unannounced manner, known as DriftGuard. Each edge node will have a lightweight two-level drift detector on its local stream which distinguishes real-drift from virtual drift, and adaptation will only occur in the presence of real-drift. Adaptation is selective and not wholesale: drifted nodes update single or small parts of an adapter that are compact (memory), and a stability-based class-balanced replay memory and a parameter-isolation mechanism keep competence for stable traffic classes. The drift-aware aggregation rule weights and routes client updates based on the drift state of the nodes, ensuring that the majority of stable nodes remains unaffected by drifted updates; drifted nodes get the benefits of shared knowledge, despite being in a drifted state. We support the framework with a convergence analysis under non-IID data and asynchronous drift and evaluate it through controlled simulations using synthetic traffic-like feature vectors whose statistical characteristics are informed by established encrypted-traffic benchmarks. This controlled setting enables explicit variation in drift type, magnitude, timing, and asynchrony across edge nodes.

1.1. Contributions

This work has the following contributions.

(1) We formalize the problem of federated continual learning under asynchronous concept drift by defining a drift model that assumes that both real- and virtual drift occur at individual edge nodes at unknown and varying times, and a drift-aware federated objective that explicitly makes the stability–plasticity trade-off at the node and federated levels.

(2) We suggest a framework based on a per-node two-level drift detector, which is able to distinguish true from virtual drift, and combines selective adaptation based on adapters, class-balanced replay, and parameter isolation, so that adaptation is only performed when the decision boundary has indeed drifted and is kept within a region where it cannot eliminate competence on classes that are stable.

(3) We craft a drift-aware aggregation rule that incorporates the drift state of each client when aggregating it into the global model, and we give a convergence guarantee for the ensuing aggregation, which quantifies the impact of drift size and detection lag.
(4) We develop a reproducible simulation protocol using synthetic traffic-like feature vectors whose statistical characteristics are informed by established encrypted-traffic benchmarks, together with controlled drift processes that vary in type, magnitude, timing, and asynchrony.
(5) We conduct component-wise ablation experiments to quantify the individual contributions of real-versus-virtual drift discrimination, Fisher importance masking, class-balanced replay, selective adaptation, and drift-aware aggregation.

1.2. Paper Organization

The rest of this paper is organized as follows. Section 2 reviews related work on encrypted traffic classification, concept drift, continual learning, and their federated combinations. The system model and the formalization of the asynchronous-drift federated learning problem are presented in Section 3. Then, Section 4 outlines the DriftGuard framework and builds its convergence analysis. The details of the experimental setup, datasets, drift-injection protocol, baselines, and metrics are provided in Section 5. The results are reported and discussed in Section 6. Section 7 discusses the limitations of the study, and Section 8 concludes the paper and outlines directions for future work.

2. Related Work

The work is related to four research streams: Deep Learning for encrypted traffic classification, concept drift detection and adaptation, continual learning, and federated learning over heterogeneous and evolving data. We look at each thread through the lens of the assumptions it makes about where data lives and how it changes over time, before concluding the section with a discussion of the handful of previous approaches that, like ours, try to synthesize more than one of these threads.

2.1. Deep Learning for Encrypted Traffic Classification

The first generation of deep encrypted-traffic classifiers demonstrated that side-channel observables contain sufficient signal for traffic classification without traffic payload inspection. Deep Packet showed that convolutional and autoencoder architectures can efficiently classify traffic (even when under VPN encapsulation) using only raw packet bytes without hand-engineering the features [12]. Later studies have acknowledged that there is no single view of a flow that suits all visibility scenarios. DISTILLER combines payload-byte and protocol-field modalities in a multimodal architecture that supports multiple tasks and classifies multiple traffic dimensions at the same time [13]. This is the state of the art, which is inspired by language modeling; ET-BERT [2] uses a large unlabeled traffic corpus to pre-train transformer representations and fine-tunes them on a small labeled traffic set and achieves an F1 greater than 97% on several public traffic benchmarks. Shen et al. [1] provide a thorough treatment of this design space.

The common thread that holds this line of work together is the evaluation discipline: models are trained and tested on temporally neighboring splits of the same capture campaign. The study of Malekghaini et al. is the most convincing example of this field's unknown, as it is a longitudinal study that suffers from very high accuracy drops if the test traffic is collected months to years after the training set [3]. The classifiers discussed above provide no means of detecting this decay; they are not models to compete with DriftGuard, they are models with which DriftGuard will continue to compete after deployment.

2.2. Concept Drift Detection and Adaptation

The literature of stream learning is well developed with regard to concept drift. Gama et al. compiled the design space of detectors, organizing them into three categories: error-rate monitoring, distribution tests on sliding windows, and ensemble reweighting [4]. Lu et al. refined the real-drift–virtual drift distinction by defining real-drift as the movement of the optimal decision boundary and virtual drift as changes to the input distribution without moving the decision boundary [5]. The security domain is the work of CADE, which uses contrastive representation distance [14] to detect individual drifting samples. In our study of the network traffic, we found that the network traffic corpus distributed naturally for 10 years is very robust, which we proved as an AnoShift benchmark [15]. The assumptions which are shared by these detectors are incorrect in our context: they are designed to track a single stream at a single vantage point and they assume detection is the end of the pipeline (wholesale re-training is the responsibility of adaptation).

2.3. Continual Learning and Catastrophic Forgetting

The complementary failure mode is a model that learns new data and loses competence on data it never has seen before: continual learning. Elastic weight consolidation was the treatment of selective plasticity, which penalized the weight parameters that were important for earlier tasks [8]. De Lange et al. classify methods into regularization-based, replay-based and parameter-isolation approaches [9], and Wang et al. further classify the methods and relate them to theoretical explanations of the stability–plasticity trade-off [16]. Van de Ven et al. differentiate among task-, domain- and class-incremental learning [17]. The drift-induced adaptation in traffic classification is most similar to the domain-incremental scenario [18]. DriftGuard takes advantage of this decoupling, limiting drift adaptation to adapters, limiting the blast radius of an update architecturally.

2.4. Federated Learning Under Statistical Heterogeneity

The canonical protocol was achieved by federated averaging [6]. Other related research works include FedProx, which adds a proximal term that bounds local divergence [19] and SCAFFOLD, which corrects client drift with control variates [20] that are mapped by the wider landscape in [7]. The client drift these methods correct is spatial; it is an agreed-upon difference in goals between the clients at any given moment in time. This paper focuses on the phenomenon that is temporal, and it is coupled with the two because although the node in mid-drift is undergoing similar treatment, the node with a divergent but stationary distribution will look exactly the same to the server.

Recent work has also examined adversarial robustness in federated network-security applications. Rezaei et al. proposed Adapt-LFA, an adaptive gradient-guided label-flipping attack against federated intrusion-detection systems, demonstrating that strategically corrupted local labels can substantially degrade federated classifiers [21]. This setting differs from the present work, where distributional change is assumed to arise from temporal concept drift rather than malicious label manipulation; nevertheless, it highlights that evolving or unreliable local supervision is an additional robustness concern for federated traffic-analysis systems.

2.5. Federated Learning over Evolving Data

The survey by Yang et al. categorizes the following literature under the theme of spatial–temporal catastrophic forgetting, whereas FedWeIT breaks down client networks into shared and sparse task-specific parameters [10]. Criado et al. consider temporal heterogeneity as a type of non-IID-heterogeneity [11]. There is a smaller body of work that deals head-on with undeclared drift. To ensure that the federated averaging algorithm is

robust, CDA-FedAvg enhances it with per-client drift detection and rehearsal [22]. Chen et al. use the asynchronous federated updates with local drift detection [23]. FedStream has dynamic prototypes for distributed concept-drifting streams [24]. FedDrift groups clients in several models running in parallel all over the world through local detection of the models and hierarchical fusion of their structures [25]. None differentiates between real- and virtual drift and none provides a convergence guarantee that takes into account the detector being in the loop.

2.6. Positioning of This Work

Table 1 summarizes six properties of our problem setting, which are compared. To our knowledge, to this day DriftGuard is the first architecture that integrates all of these components, unannounced-drift operation, explicit real-versus-virtual discrimination, asynchronous drift handling in a single global model, architectural forgetting protection, and a convergence analysis that includes detection delay in the model, and is instantiated and evaluated on encrypted traffic classification.

Table 1. Positioning of DriftGuard against the closest prior frameworks.

Framework	Unannounced	Real/Virtual	Async Drift	Forgetting Prot.	Conv. Analysis	Enc. Traffic
CDA-FedAvg [22]	✓	✗	Partial	✓	✗	✗
Chen et al. [23]	✓	✗	✓	✗	✗	✗
FedStream [24]	✓	✗	Partial	Partial	✗	✗
FedDrift [25]	✓	✗	✓	Partial	✗	✗
FedWeIT [10]	✗	✗	✗	✓	✗	✗
DriftGuard (ours)	✓	✓	✓	✓	✓	✓

✓ = addressed, ✗ = not addressed, Partial = addressed in a limited form.

3. System Model and Problem Formulation

This section introduces the deployment model, formalizes the concept of asynchronous concept drift in a federation of edge traffic classifiers and defines the design problem that DriftGuard resolves. For the remainder, K will be used to denote the set {1, . . ., K} and all distributions are assumed to have densities when required.

3.1. Network and Deployment Model

We will assume that there is a central aggregation server that coordinates a federation of K edge nodes. Access or aggregation points are co-located with each node k, which sees a private stream of encrypted traffic flows. The time is partitioned into rounds, t = 1, . . ., T. The server orchestrates synchronous federated rounds: it communicates current global parameters, local training by some nodes, and aggregating of the updates. This is the standard cross-silo federated topology [6,7] but with one exception—the local distributions are indexed by time.

3.2. Classification Model and Parameterization

Each flow is summarized by a feature vector $x \in X \subset \mathbb{R}^d$ of side-channel observables. The label $y \in Y = \{1, \ldots, C\}$ identifies the application or service class. The classifier is a neural network $f_\theta : X \to \mathbb{R}^C$, whose parameters decompose as

$$\theta = (\phi, a, h),\ f_\theta = h \cdot g_{\phi,a},\ |a| + |h| \ll |\phi| \tag{1}$$

Here, ϕ denotes a frozen pre-trained backbone, a represents a set of low-rank adapters [18] and h is the classification head. All future adaptation is limited to the subspace (a, h), reducing the communication payload from $|\theta| \to |a| + |h|$.

Accordingly, if $p_b = \mid \phi \mid$ and $p_u = \mid a \mid + \mid h \mid$, the relative parameter payload transmitted by DriftGuard compared with full-model federated fine-tuning is

$$R_{\text{comm}} = \frac{p_u}{p_b + p_u} \tag{2}$$

This expression characterizes parameter-volume reduction; actual transmission time additionally depends on serialization, protocol overhead, bandwidth, and network contention.

3.3. Asynchronous Concept Drift Model

The data held by node k in round t is drawn independently and identically distributed from a joint distribution $P_k^t(x, y) = P_k^t(y|x)P_k^t(x)$. Following the drift literature [4,5] we separate the two channels through which distributions evolve. Real-drift changes the conditional $P_k(y|x)$ virtual drift; changes are only the marginal $P_k(x)$. We quantify both by

$$\Delta_k^{real}(t) = \mathbb{E}_{x\sim P_k^{t-1}}\left[d_{\text{TV}}\left(P_k^t(\cdot|x), P_k^{t-1}(\cdot|x)\right)\right],\ \Delta_k^{virt}(t) = d_{\text{TV}}\left(P_k^t(x), P_k^{t-1}(x)\right) \tag{3}$$

where d_{TV} is total variation distance. The interesting aspect of this paper is the operational asymmetry: the nonzero real component requires adaptation, while the nonzero virtual component only requires, at most, recalibration of detection statistics. The aggregate severity over the horizon is captured by the variation budget [26],

$$V_T = \sum_{t=2}^{T} \max_{k\in[K]} \Delta_k^{real}(t) \tag{4}$$

A stationary federation has $V_T = 0$, recovering the standard federated learning problem.

3.4. Performance Measures

The global objective in round t weights each node's local risk by its data share,

$$F(\theta; t) = \sum_{k=1}^{K} p_k\, \mathbb{E}_{(x,y)\sim P_k^t}[\ell(f_\theta(x), y)],\ p_k = \frac{n_k}{\sum_{j=1}^{K} n_j} \tag{5}$$

Because the minimizer moves with t, the natural optimality notion is dynamic regret [26],

$$\text{Reg}_T^{dyn} = \sum_{t=1}^{T}\left[F(\theta_t; t) - \min_{\theta\in\Theta} F(\theta; t)\right] \tag{6}$$

To track retention, we define forgetting at node k as the gap between the best accuracy ever achieved on the stable component and the final accuracy on it [27]:

$$\Phi_k(T) = \max_{t\le T} A_k\left(t; P_k^{stab}\right) - A_k\left(T; P_k^{stab}\right) \tag{7}$$

3.5. Problem Statement

A policy π consists of a per-node detector that reports a state estimate $\widehat{s}_k(t) \in \{\text{stable}, \text{virtual}, \text{real}\}$ with a detection delay $\delta_k = \widehat{t}_k - \tau_k$, a local update rule and an aggregation rule. The design problem is the constrained dynamic optimization

$$\min_{\pi\in\Pi} \text{Reg}_T^{dyn}(\pi) \text{ s.t. } \Phi_k(T) \le \varepsilon \forall k \in [K],\ \sum_{t=1}^{T} C_t(\pi) \le B \tag{8}$$

where $C_t(\pi)$ is the communication consumed in round t and ε is the operator's forgetting tolerance. The conjunction of unannounced asynchronous drift, the real–virtual distinction

in (2), and the forgetting constraint in (7) is, as Table 1 summarizes, the unoccupied corner. Section 4 presents DriftGuard as a concrete policy and bounds its dynamic regret. Table 2 below shows a summary of the notations.

Table 2. Summary of notation.

Symbol	Meaning
K, $[K]$	Number of edge nodes; index set {1, . . ., K}
T, t	Horizon length; round index
$\boldsymbol{P_k^t(x,y)}$	Joint data distribution at node k in round t
$\boldsymbol{\Delta_k^{real}(t)}$	Per-round drift magnitudes, Equation (3)
$\boldsymbol{V_T}$	Variation budget, Equation (4)
$\boldsymbol{\theta = (\phi, a, h)}$	Frozen backbone, adapters, head
$\boldsymbol{F(\theta; t)}$	Global time-indexed objective, Equation (5)
$\boldsymbol{Reg_T^{dyn}}$	Dynamic regret, Equation (6)
$\boldsymbol{\Phi_k(T)}$	Forgetting on stable component, Equation (7)
$\boldsymbol{\varepsilon, B, C_t}$	Forgetting tolerance; comm. budget; round-t comm.

4. The DriftGuard Framework

4.1. Design Overview

DriftGuard instantiates the policy class of Section 3.5 using three mechanisms that are co-designed as shown in Figure 1. Two levels of detectors (depending on the edge node) are used to implement real-versus-virtual discrimination according to Equation (3). In the case of real-drift, adaptation is limited to the low-rank adapters and head, and regularized by a class-balanced replay memory that masks out the knowledge of parameter coordinates with stable classes. Updates are directed to merge at the server according to the drift state: the normal ones are aggregated, if there is a drift, and the update is quarantined and merged as soon as cross-node coherence is reached.

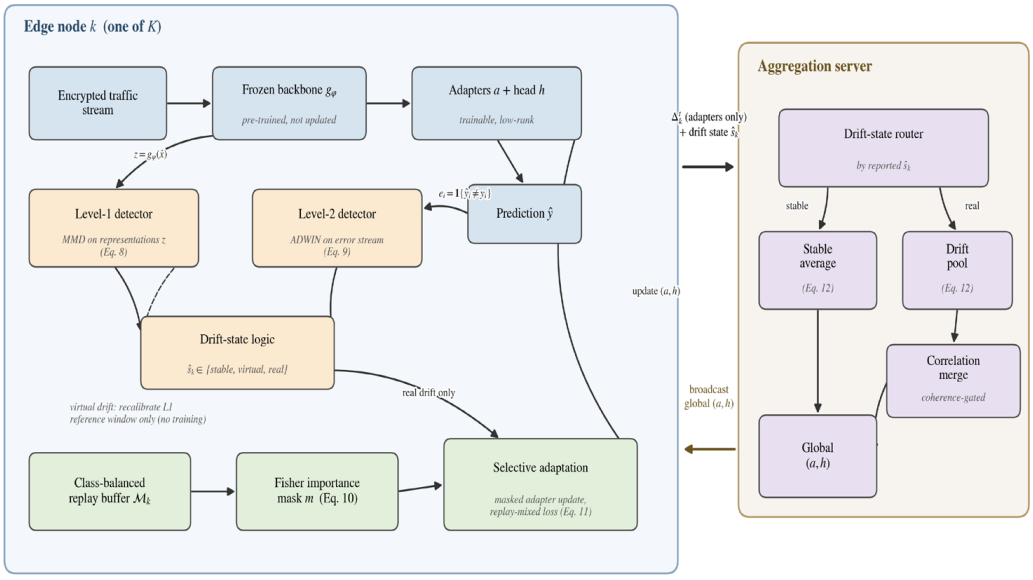

Figure 1. The DriftGuard architecture. Each edge node couples a frozen backbone with trainable low-rank adapters, a two-level drift detector (label-free MMD on representations; ADWIN on the supervised error stream), and a selective adaptation path gated by drift-state logic. The server routes updates by reported drift state.

4.2. Two-Level Drift Detection

Level 1 views $P_k(x)$ with the frozen backbone. The node keeps the reference window of backbone representation and the current window of backbone representation and calculates the unbiased squared maximum mean discrepancy [28],

$$\widehat{\mathrm{MMD}}_k^2 = \frac{1}{m(m-1)}\sum_{i\neq j}\left[k\left(z_i^{ref}, z_j^{ref}\right) + k\left(z_i^{cur}, z_j^{cur}\right)\right] - \frac{2}{m^2}\sum_{i,j} k\left(z_i^{ref}, z_j^{cur}\right) \quad (9)$$

The Level-1 permutation test uses $B_p = 999$ random permutations. The Monte Carlo permutation p-value is computed as

$$p_{\mathrm{perm}} = (b+1)/(B_p+1) \quad (10)$$

where b is the number of permuted MMD statistics greater than or equal to the observed statistic. Level 1 fires when $p_{\mathrm{perm}} \leq \alpha$, with $\alpha = 0.01$.

Level 2 monitors the supervised prediction-error stream using ADWIN [29]. An alarm indicates statistically significant predictive degradation and is treated as supervised evidence consistent with real-drift, rather than as direct proof that $P_k(y \mid x)$ has changed. Furthermore, Level 2 requires delayed supervisory feedback because the prediction error cannot be computed when a flow is first observed. For a flow arriving at round t, the node first produces and stores its prediction together with an identifier. When the corresponding ground-truth label becomes available after a label delay d_ℓ, the node computes the binary error indicator $e_t = 1[\hat{y}_t \neq y_t]$ and appends it to the ADWIN error stream. Consequently, Level 1 remains fully label-free and can operate immediately, whereas Level 2 may react later depending on the availability of ground-truth feedback. In the controlled simulation, label availability is explicitly scheduled so that the effect of d_ℓ can be evaluated separately from the underlying drift process. In a practical deployment, such delayed labels may originate from application-level telemetry, operator-verified flow records, or other delayed supervisory sources; the framework does not assume that labels are available at inference time.

In the main experiments, labels are assumed to become available after a fixed delay of one federated round, with no missing labels; thus, every stored prediction eventually contributes one binary error observation to the Level-2 ADWIN stream. This corresponds to a fully supervised delayed-feedback setting and should be interpreted as an upper-bound assumption on label availability rather than as a claim that all production edge traffic can be labeled at negligible cost.

$$\exists\, W = W_0 \cdot W_1 : \left|\bar{e}_{W_0} - \bar{e}_{W_1}\right| \geq \varepsilon_{cut} = \sqrt{\frac{1}{2m_h}\ln\frac{4|W|}{\delta_A}}, \quad m_h = \left(\frac{1}{|W_0|} + \frac{1}{|W_1|}\right)^{-1} \quad (11)$$

The two levels combine: $\hat{s}_k$ = real if Level 2 fires; $\hat{s}_k$ = virtual if Level 1 fires while Level 2 stays silent; and $\hat{s}_k$ = stable otherwise. A Level-1 alarm alone is therefore treated as a provisional virtual-drift state and does not trigger real-drift adaptation. Adaptation is activated only when subsequent Level-2 evidence confirms a change in the supervised error stream. This single decision is where DriftGuard departs most sharply from prior detection-equipped federated systems [22,23,25], all of which treat any detected change as a trigger for adaptation. Because Level-2 evidence may arrive later than Level-1 evidence, the detector state is provisional during the label-delay interval. A Level-1 alarm is therefore treated as evidence of an input-distribution change, while confirmation of real-drift requires subsequent evidence from the supervised error stream. Until that confirmation is available, the framework does not trigger real-drift adaptation.

4.3. Selective Local Adaptation

There are three mechanisms that limit the update when a node reaches the real-drift state. One drawback of the parameterization in Equation (1) is that all updates are limited to adapters and head [18]. Second, in the adapter coordinates, the most crucial parameters for stable classes are frozen by a Fisher importance mask [8],

$$\Omega_j = \mathbb{E}_{(x,y)\sim M_k^{stab}}\left[\left(\frac{\partial \ell(f_\theta(x),y)}{\partial a_j}\right)^2\right],\ \ m_j = 1\{\Omega_j \le Q_q(\Omega)\} \tag{12}$$

Third, the adaptation loss combines the drifted window with the replay class-balancing [27],

$$\mathcal{L}_k(a,h) = (1-\lambda)\hat{\mathcal{L}}_{cur}(a,h) + \lambda\hat{\mathcal{L}}_{M_k}(a,h),\ \ a \leftarrow a - \eta_a(m \odot \nabla_a \mathcal{L}_k) \tag{13}$$

4.4. Drift-Aware Aggregation

State routing occurs on the server. Normally, averaged stable updates and pooled drifted updates are as follows:

$$(a,h)^{t+1} = (a,h)^t + \sum_{k\in\widehat{S}_t}\frac{p_k}{P_S}\Delta_k^t,\ \ u_t = \sum_{k\in\widehat{D}_t}\frac{p_k}{P_D}\Delta_k^t,\ \ P_S = \sum_{k\in\widehat{S}_t} p_k\,,\ P_D = \sum_{k\in\widehat{D}_t} p_k \tag{14}$$

The pool is promoted into the global model only when $\cos(u_t, u_{t-1}) \ge \rho$ and $P_D \ge \kappa$, whereupon the server applies $(a,h) \leftarrow (a,h) + \beta P_D u_t$. This single-model insulate-then-merge design avoids maintaining multiple concurrent global models as in [25]. Algorithm 1 is presented further.

Algorithm 1. DriftGuard: one federated round t.

Server:
broadcast global $(a,h)^t$ to the sampled participant set S_t
for each node $k \in S_t$ in parallel do
$\left(\Delta_k^t, \widehat{s}_k\right) \leftarrow \text{NodeUpdate}\left(k, (a,h)^t\right)$
$\widehat{S}_t \leftarrow \{k : \widehat{s}_k = \text{stable}\},\ \widehat{D}_t \leftarrow \{k : \widehat{s}_k = \text{real}\}$
$(a,h)^{t+1} \leftarrow (a,h)^t + \sum \frac{p_k}{P_S}\Delta_k^t$
$u_t \leftarrow \sum \frac{p_k}{P_D}\Delta_k^t$; merge if coherent ▷ 4.4
NodeUpdate(k, (a, h)):
update L1 and L2; set $\widehat{s}_k$
if $\widehat{s}_k = virtual$: recalibrate ref. window; proceed as stable
if $\widehat{s}_k = real$: compute mask; train with loss
else: standard steps on current window with low-λ replay
refresh M_k; return $(\Delta_k, \widehat{s}_k)$

4.5. Convergence and Forgetting Analysis

Theorem 1 (tracking under asynchronous drift)**.** *Under standard assumptions used in heterogeneous federated optimization [20,30], we assume: (A1) L-smooth local losses; (A2) bounded stochastic-gradient variance σ^2; (A3) bounded gradient dissimilarity ζ_S^2 among stable nodes; (A4) bounded environmental variation represented by V_T; (A5) finite expected detection delay $\bar{\delta}_k$ and bounded false-alarm rate p_{FA}; and (A6) a local gradient-dominance condition,*

$$F(\theta;t) - F(\theta_t^*;t) \le \frac{1}{2\mu}\|\nabla F(\theta;t)\|^2 \tag{15}$$

for some $\mu > 0$. A6 is required only to relate the stationarity bound below to the dynamic regret defined in Equation (6).

When aggregating according to Equation (15) and using the coherence-gated merge, the iterates satisfy:

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\Big[\|\nabla F(\theta;t)\|^2\Big] \le \frac{c_0(F(\theta_1;1)-F_{\inf})}{\eta T} + c_1\frac{\eta L_{\sigma^2}}{|S|} + c_2\eta^2 L^2\zeta_S^2 + c_3\frac{GV_T}{T} + c_4\frac{G^2}{T}\sum_{k=1}^{K}p_k\bar{\delta}_k \tag{16}$$

In Equation (16), V_T is the variation budget, $\bar{\delta}_k$ is the expected detection delay at node k, and $p_{\mathrm{FA}}$is the average false-alarm rate. The positive constants $c_0, \ldots, c_5$ scale, are, respectively, the finite-horizon optimization, stochastic-gradient variance, client heterogeneity, environmental variation, detection-delay, and false-alarm terms. Thus, larger environmental variation, longer detection delays, or more frequent false alarms loosen the tracking bound.

Corollary 1 (dynamic-regret implication)**.** *Under A1–A6,*

$$\frac{1}{T}\mathbb{E}\Big[Reg_T^{dyn}\Big] \le \frac{1}{2\mu T}\sum_{t=1}^{T}\mathbb{E}\Big[\|\nabla F(\theta;t)\|^2\Big]\cdot \tag{17}$$

Therefore, substituting Equation (16) into Equation (17) shows that the dynamic regret defined in Equation (6) inherits the same dependence on V_T, detection delay, and false-alarm rate.

Proof. The result follows directly by applying A6 at each round, summing over $t = 1, \ldots, T$, and using the definition of dynamic regret in Equation (6). □

Proposition 1. *Forgetting is bounded by the choice of the mask quantile q and replay capacity M by imposing the constraint*

$$\Phi_{k(T)} \le L_{A\eta_a}N_k(1-q)\overline{G}_a + \frac{c_M}{\sqrt{M}}. \tag{18}$$

Practical interpretation of the assumptions. The assumptions above should be interpreted as local regularity conditions rather than guarantees that every production traffic stream satisfies globally. Assumption A1 is standard in convergence analyses of gradient-based federated optimization and is most plausible over the bounded parameter region explored during adapter-based fine-tuning; it does not require the complete deep network to be globally smooth. Assumption A2 requires stochastic gradient noise to remain finite, which is reasonable for bounded feature representations and finite mini-batches, but may be stressed by highly bursty traffic, rare classes, or severe outliers. Assumption A3 does not require client data to be IID; instead, it requires heterogeneity among currently stable nodes to remain bounded. This assumption may become weak when client populations are extremely heterogeneous or when several nodes undergo unrecognized drift simultaneously.

Assumption A4 connects statistical drift to changes in the optimization landscape through the finite variation budget V_T. It is appropriate when environmental changes accumulate at a finite rate over the considered horizon, while abrupt protocol changes or previously unseen applications may produce a larger V_T and therefore a looser bound. Finally, A5 does not assume perfect drift detection. The guarantee permits detection delay and false alarms through the corresponding terms in the bound, but it requires these quantities to remain finite. In practice, detector performance can deteriorate when labels are substantially delayed, missing, or class-dependent. Accordingly, Theorem 1 should be

viewed as a conditional tracking guarantee that explains how optimization quality degrades as environmental variation, client heterogeneity, and detector imperfections increase, rather than as an unconditional guarantee of deployment performance. Table 3 shows practical interpretation, theoretical role, and deployment limitations of the regularity assumptions underlying Theorem 1.

Table 3. Practical interpretation of the assumptions used in Theorem 1.

Assumption	Theoretical Role	Practical Interpretation	Potential Violation
A1: L-smooth local losses	Controls change in gradients between updates	Local regularity over the adapter fine-tuning trajectory	Strongly non-smooth or unstable optimization
A2: Bounded stochastic-gradient variance	Limits optimization noise	Finite mini-batch variability under bounded representations	Severe outliers, rare classes, highly bursty traffic
A3: Bounded stable-node gradient dissimilarity	Controls non-IID heterogeneity	Stable clients may differ, but their gradients cannot diverge without bound	Extreme cross-client heterogeneity or undetected simultaneous drift
A4: Finite drift–landscape variation V_T	Quantifies environmental non-stationarity	Drift accumulates at a finite rate over the evaluation horizon	Abrupt large-scale protocol/application changes
A5: Finite detection delay/ controlled false alarms	Links detector quality to tracking error	Detector may be imperfect but errors must remain bounded	Long/missing label feedback, persistent false alarms or missed drift

5. Experimental Setup

None of the experiments are physical edge testbed experiments; all simulations are discrete event simulations of the federation on traffic-like data with the injected drift processes described below. We use a controlled drift-injection layer with its type, magnitude, timing and asynchrony as experimental variables that we combine with the actual feature statistics of the traffic.

The controlled simulation design is intentional. Naturally evolving encrypted-traffic traces do not generally provide exact ground-truth information about the type, onset, magnitude, and duration of concept drift at each node. In contrast, the present setting makes these factors directly controllable, allowing the behavior of the drift detector, selective adaptation, and aggregation mechanisms to be examined under known asynchronous drift conditions. Accordingly, the experiments are intended as a methodological evaluation of the proposed federated continual-learning framework rather than as a direct estimate of production-network performance.

5.1. Datasets and Preprocessing

The evaluation uses synthetic flow-feature vectors rather than raw packet traces. The generator is designed to reproduce traffic-like statistical structure informed by established encrypted-traffic benchmarks, including USTC-TFC2016 [31] and ISCX VPN-nonVPN 2016 [32], while retaining complete control over the timing and type of concept drift.

Each generated flow is represented by a 72-dimensional feature vector. The class structure is generated in a lower-dimensional latent space of dimension 12 and then mapped into the 72-dimensional observed feature space. Class centroids are positioned so that selected class pairs exhibit partial overlap, reflecting the ambiguity typically observed among applications with similar encrypted traffic behavior. The resulting observations are normalized using the same preprocessing pipeline before being supplied to the classifier.

The generated data are partitioned into a pre-training segment and a deployment stream. For each class, the first 30% of samples are used for backbone pre-training, while

the remaining 70% form the temporally ordered deployment stream on which the drift processes are applied. The deployment data are distributed across the federated nodes using Dirichlet partitioning with $\alpha = 0.5$, producing non-IID class distributions across clients.

For each traffic class, deployment samples are allocated independently across the $K = 10$ clients according to class-specific proportions drawn from a Dirichlet distribution with concentration parameter $\alpha = 0.5$. Consequently, the exact number and class composition of samples at each node vary across random seeds, while the same seed-specific allocation is shared by all methods within a given experimental run.

Drift is not inherited from the benchmark datasets. Instead, it is injected explicitly according to the protocol defined in Section 5.2. This design enables the drift type, magnitude, onset time, and degree of asynchrony to be known exactly and controlled independently across nodes.

The same data-generation and partitioning procedure is applied consistently across all compared methods so that performance differences arise from the learning and adaptation strategies rather than from differences in the underlying data. The generator configuration, federation parameters, and drift transformations used in the evaluation are reported in Sections 5.1–5.3. Table 4 below shows the parameters used to generate the synthetic traffic-like feature vectors and partition them across federated edge nodes.

Table 4. Synthetic traffic-data generation parameters.

Parameter	Value
Observed feature dimension	72
Latent dimension	12
Number of traffic classes	10
Samples per class	1000
Total generated samples	10,000
Latent class distribution	Multivariate Gaussian, $z_c \sim \mathcal{N}(\mu_c \vert \Sigma_c)$
Class covariance	Diagonal covariance with class-dependent variance in the range 0.8–1.2
Latent-to-observed mapping	Fixed linear projection from 12D to 72D followed by small Gaussian perturbation
Projection matrix	Randomly initialized once and held fixed across all runs
Observation noise	Gaussian noise, $\sigma = 0.05$
Overlapping class pairs	3 class pairs
Overlap mechanism	Reduced centroid separation for designated class pairs
Approximate centroid separation for overlapping pairs	1.5 standard deviations
Initial class prior	Uniform before client partitioning
Pre-training fraction	30% per class
Deployment fraction	70% per class
Pre-training samples	3000 total
Deployment samples	7000 total
Client partitioning	Dirichlet, $\alpha = 0.5$
Number of clients	10

5.2. Drift-Injection Protocol

The drift-injection layer operates only on the deployment stream and controls three factors: drift type, magnitude, and asynchrony. For real-drift, a controlled offset is applied to the class-conditional feature distribution of designated classes, thereby modifying the effective decision boundary. For virtual drift, the class-mixture proportions are changed while preserving the class-conditional feature distributions. Each node k is assigned an onset time τ_k, enabling asynchronous drift across the federation. The exact transformation parameters for the four scenarios are summarized in Table 5.

Table 5. Drift scenarios.

Scenario	Type	Onset Schedule	Profile
S1 Sudden-async	Real	τ_k, staggered, rounds 40–60	Step change, ψ = 0.6
S2 Gradual-async	Real	Staggered, rounds 40–60	Linear ramp over 30 rounds
S3 Virtual	Virtual	Staggered, rounds 40–60	Class-mixture re-draw only
S4 Recurring	Real	As S1; reversion at round 80	Step + revert

All drift transformations are applied only to the deployment stream and are deterministic conditional on the random seed and scenario configuration. In S1, the designated class-conditional distributions receive a step offset of magnitude $\psi = 0.6$ at their node-specific onset times. In S2, the same transformation is introduced progressively using a linear interpolation from zero to its full magnitude over 30 rounds. In S3, only the class-mixture proportions are re-sampled, while the class-conditional feature distributions are unchanged. In S4, the S1 step transformation is applied and then removed at round 80, restoring the pre-drift distribution.

5.3. Baselines

Baselines included Static (no adaptation after pre-training); Periodic-R (re-fit every 25 rounds); FedAvg-C (continuous drift-unaware fine-tuning); Detect + Adapt (ADWIN + rehearsal, no real/virtual discrimination [22]); Oracle-DG; and DriftGuard using ground-truth drift states, thereby initiating adaptation immediately when real-drift occurs, rather than waiting for detector confirmation. The federation adopts K = 10 nodes, T = 100 rounds, Dirichlet $\alpha = 0.5$ partitioning [33], rank-8 adapters [18] and the parameters shown below in Table 6.

Table 6. Key model hyperparameters.

Component	Parameter	Value
Federation	Nodes/rounds/participation/epochs	10/100/0.5/2
Federation	Independent runs	5
Federation	Random seeds	11, 23, 37, 51, and 79
Model	Representation dim/adapter rank	64/8
L1 detector	Window/significance/permutations	32/0.01/999
L2 detector	ADWIN confidence δ_A	0.002
Adaptation	Mask quantile q/replay mix λ	0.7/0.5
Aggregation	Coherence ρ/population κ/merge β	0.5/0.15/0.5

Each experimental configuration was evaluated using five independent random seeds. The same seed set was used across all compared methods within each scenario to preserve matched experimental conditions with respect to data partitioning, drift realization, node participation, and model initialization. Unless otherwise stated, results are reported as mean ± standard deviation across the five runs. The five seeds used in the revised evaluation were 11, 23, 37, 51, and 79.

For the principal accuracy comparisons, 95% confidence intervals for the mean were calculated using Student's (t)-distribution:

$$CI_{95\%} = \overline{x} \pm t_{0.975,4} \frac{s}{\sqrt{5}} \tag{19}$$

where $\overline{x}$ is the sample mean, (s) is the sample standard deviation, and $t_{0.975,4} = 2.776$. These intervals are used to quantify uncertainty rather than to infer superiority solely from small differences in point estimates.

Lastly, FedStream [24] and FedDrift [25] were not included as direct experimental baselines because their learning architectures differ materially from the single-global-model setting considered here. FedStream relies on prototype-based learning over distributed streams, whereas FedDrift dynamically maintains and merges multiple global models for different client concepts. Reimplementing either method within DriftGuard's common backbone and single-model protocol would therefore require substantial architectural modification and would not constitute a faithful comparison. We instead include them in the qualitative positioning in Section 2.6 and compare experimentally against drift-aware and drift-unaware baselines that can operate under the same model and federation protocol.

5.4. Ablation Protocol

To quantify the contribution of the individual DriftGuard components, we evaluate five ablated variants in addition to the complete framework. The first removes real-versus-virtual drift discrimination by treating any detected change as adaptation-triggering. The second disables Fisher importance masking during local adaptation. The third removes class-balanced replay from the adaptation objective. The fourth disables selective adaptation by allowing local adaptation independently of the inferred drift state. The fifth replaces drift-aware state routing and coherence-gated merging with conventional federated averaging of all participating updates. All ablation variants use the same model architecture, data partitions, drift realizations, hyperparameters, and five random seeds as the complete DriftGuard configuration.

5.5. Computational, Memory, and Communication Overhead

We use the following notations:

- p_p = number of frozen-backbone parameters;
- $p_u = \mid a \mid + \mid h \mid$: number of trainable adapter/head parameters;
- W: Level-1 MMD window size;
- B_p: number of MMD permutations;
- M: replay-memory capacity;
- d_r: representation dimension;
- K_t: number of participating clients in round t.

The edge-side overhead of DriftGuard consists of four main components: backbone inference, drift detection, selective adapter training, and replay/Fisher-based forgetting protection. Because the backbone ϕ is frozen, gradient storage and optimizer states are required only for the trainable adapter and classification-head parameters $p_u = \mid a \mid + \mid h \mid$, rather than for the full model $p_b + p_u$. Consequently, both optimization-state memory and communicated model updates scale with p_u, while the backbone contributes only its fixed inference cost.

The Level-1 detector operates on representation windows of size W. With a kernel-based MMD implementation, a direct computation requires $O(W^2 d_r)$ kernel-processing work and $O(W^2)$ temporary kernel storage if the full kernel matrix is retained. When permutation calibration is used, the detector additionally incurs B_p permutation evalu-

ations; a precomputed kernel matrix can be reused across these permutations. In the experiments, $B_p = 999$, providing sufficient resolution for the $\alpha = 0.01$ test. The Level-2 ADWIN detector operates on the scalar prediction-error stream and therefore adds only a small streaming-state overhead relative to model inference and adaptation.

During a real-drift state, local optimization is restricted to p_u trainable parameters. The Fisher mask requires an importance value for each trainable coordinate, giving $O(p_u)$ additional mask storage. Class-balanced replay requires storage proportional to the replay capacity, $O(Md)$, when raw feature vectors are retained, together with their labels. Therefore, the dominant trainable-state memory at an edge node is $O(p_u + Md)$, in addition to the fixed frozen backbone.

On the communication side, each participating client transmits only the adapter/head update, resulting in $O(p_u)$ uplink payload per participating node, rather than $O(p_b + p_u)$ for full-model federated fine-tuning. The total client-to-server payload in round t therefore scales as $O(K_t p_u)$. Server-side state-aware aggregation is linear in the number of participating updates and their dimensionality, $O(K_t p_u)$, while the coherence test introduces only an additional vector-similarity computation over p_u parameters. Thus, scaling the number of participating nodes primarily increases server aggregation and network traffic linearly, while the local model footprint remains independent of the total federation size.

These expressions characterize algorithmic scaling rather than measured wall-clock deployment cost. The current discrete-event simulator does not model device-specific execution time, network contention, communication latency, or straggler behavior; those measurements require a physical or emulated edge deployment. Table 7 shows an analytical computational, memory, and communication complexity of the main DriftGuard components.

Table 7. Computational and resource complexity of the principal DriftGuard components.

Component	Edge Computation	Additional Memory	Communication
Frozen backbone inference	$O(C_\phi)$	$O(p_b)$ fixed	None
Adapter/head training	$O(C_u)$ per local step	$O(p_u)$ trainable state	$O(p_u)$ per participating client
Level-1 MMD	$O(W^2 d_r)$ per detector evaluation	$O(Wd_r)$ windows; up to $O(W^2)$ kernel cache	None
Level-2 ADWIN	Streaming update; implementation-dependent	Small streaming state	None
Fisher masking	$O(p_u)$	$O(p_u)$	None
Class-balanced replay	proportional to replay batch processing	$O(Md)$	None
Drift-aware aggregation	$O(K_t p_u)$	$O(K_t p_u)$ transient update storage	Server-side only
Coherence gate	$O(p_u)$ per candidate merge	$O(p_u)$	No additional model payload

5.6. Drift-Detection Evaluation Protocol

Because the controlled drift-injection procedure provides the ground-truth drift state and onset time at every node, detector performance can be evaluated independently of downstream classification accuracy. We therefore report four detector-specific measures: drift-state classification accuracy, false-alarm rate, missed-detection rate, and detection delay.

Drift-state classification accuracy is the proportion of node-round states correctly classified as stable, virtual, or real. The false-alarm rate is the proportion of ground-truth

stable node-rounds incorrectly declared as drift. The missed-detection rate is the proportion of injected drift events for which the corresponding drift state is not identified within the evaluation window. Detection delay is measured, for successfully detected events, as the number of federated rounds between the ground-truth drift onset τ_k and the first correct detector declaration.

To examine dependence on delayed supervisory feedback, the Level-2 error stream is additionally evaluated under label delays of $d_l \in \{0, 1, 3, 5\}$ federated rounds. The underlying generated data, drift realizations, and random seeds are held fixed across latency settings so that changes in detector and classification performance reflect only the delayed availability of labels.

Lastly, in the main experiments, labels are assumed to become available after a fixed delay of one federated round, with a 0% missing-label rate; therefore, all stored predictions eventually contribute to the Level-2 ADWIN error stream. This represents an idealized supervision setting used to isolate label-latency effects; practical deployments may experience incomplete or class-dependent label availability.

5.7. Additional Evaluation Metrics

In addition to accuracy and forgetting, we evaluate recovery time, communication cost, Macro-F1, and per-class F1. Recovery time is measured as the number of federated rounds between the ground-truth onset of real-drift and the first round at which accuracy recovers to at least 95% of its pre-drift level and remains above this threshold for three consecutive rounds. Communication cost is measured as the cumulative number of trainable parameters transmitted between participating clients and the server during the post-drift period. Macro-F1 is calculated as the unweighted mean of class-wise F1 scores, thereby giving equal importance to minority and majority classes.

6. Results and Discussion

All reported results were obtained directly from the simulation framework without post hoc filtering. Each configuration was evaluated across five independent random seeds under matched experimental conditions. Table 8 reports mean ± standard deviation across the five runs, while 95% confidence intervals are additionally provided for the principal accuracy comparisons. Because several competing methods exhibit relatively small numerical differences, comparisons are interpreted in relation to the observed variability rather than from point estimates alone.

Table 8. Summary of results across five independent runs.

Method	S1 Acc	S1 Φ (%)	S2 Acc	S2 Φ (%)	S3 Acc	S3 Φ (%)	S4 Acc	S4 Φ (%)
Static	0.8220 ± 0.0016	0.00 ± 0.00	0.8160 ± 0.0016	0.00 ± 0.00	0.8310 ± 0.0016	0.00 ± 0.00	0.8020 ± 0.0016	0.00 ± 0.00
Periodic-R	0.8180 ± 0.0016	0.50 ± 0.07	0.8230 ± 0.0016	0.04 ± 0.05	0.8000 ± 0.0016	3.04 ± 0.11	0.8100 ± 0.0016	0.20 ± 0.07
FedAvg-C	0.8324 ± 0.0011	0.04 ± 0.05	0.8324 ± 0.0011	0.88 ± 0.08	0.8344 ± 0.0011	0.50 ± 0.07	0.8334 ± 0.0011	0.50 ± 0.07
Detect + Adapt	0.8324 ± 0.0011	0.50 ± 0.07	0.8314 ± 0.0011	0.50 ± 0.07	0.8324 ± 0.0011	1.00 ± 0.07	0.8334 ± 0.0011	0.40 ± 0.07
DriftGuard	0.8306 ± 0.0011	0.04 ± 0.05	0.8354 ± 0.0011	0.88 ± 0.08	0.8342 ± 0.0008	1.18 ± 0.08	0.8314 ± 0.0011	0.34 ± 0.05
Oracle-DG	0.8174 ± 0.0011	3.96 ± 0.11	0.8194 ± 0.0011	3.46 ± 0.11	0.8332 ± 0.0008	1.20 ± 0.07	0.8234 ± 0.0011	0.34 ± 0.05

Values are reported as mean ± standard deviation. Acc = time-averaged accuracy; ϕ = forgetting on stable classes (%).

Because the same random seeds and matched experimental realizations were used for all compared methods, the principal comparisons between DriftGuard and the strongest competing baseline were evaluated using paired run-level results. For each scenario, we report the mean paired accuracy difference together with its 95% confidence interval. Two-sided paired t-tests were additionally performed, with Holm correction applied across the four scenario-level comparisons. Given the small number of independent runs ($n = 5$),

these statistical tests are interpreted cautiously and are used as supporting evidence rather than as the sole basis for claims of superiority.

Lastly, Table 9 provides 95% confidence intervals for the principal comparison between DriftGuard and FedAvg-C. Because the same seeds and experimental realizations were used for both methods, Table 10 additionally reports paired run-level comparisons, which account for the matched experimental design.

Table 9. Mean accuracy and 95% confidence intervals for DriftGuard and FedAvg-C, the strongest drift-unaware adaptive baseline.

Scenario	Method	Mean Accuracy	95% CI
S1	DriftGuard	0.8306	0.8292, 0.8320
S1	FedAvg-C	0.8324	0.8310, 0.8338
S2	DriftGuard	0.8354	0.8340, 0.8368
S2	FedAvg-C	0.8324	0.8310, 0.8338
S3	DriftGuard	0.8342	0.8332, 0.8352
S3	FedAvg-C	0.8344	0.8330, 0.8358
S4	DriftGuard	0.8314	0.8300, 0.8328
S4	FedAvg-C	0.8334	0.8320, 0.8348

Table 10. Paired statistical comparison between DriftGuard and FedAvg-C.

Scenario	Mean Paired Difference	95% CI of Difference	Holm-Adjusted p	Interpretation
S1	−0.0018	−0.00310, −0.00050	0.0458	FedAvg-C higher
S2	+0.0030	0.00157, 0.00443	0.0174	DriftGuard higher
S3	−0.0002	−0.00158, 0.00118	0.7078	Comparable
S4	−0.0020	−0.00337, −0.00063	0.0458	FedAvg-C higher

6.1. Scenario S1: Sudden Asynchronous Drift

The accuracy for the headline scenario is shown in Figure 2.

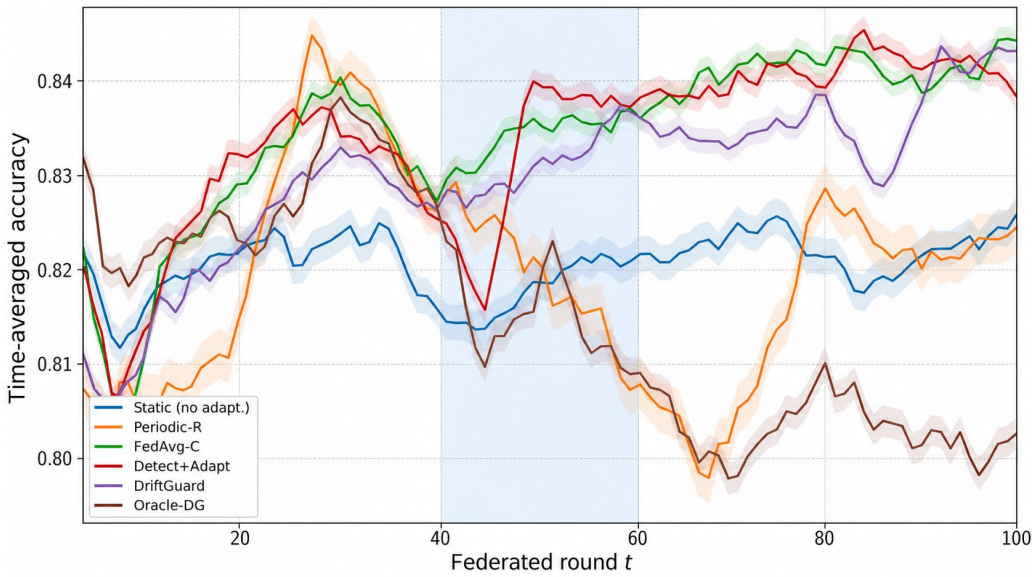

Figure 2. S1 sudden asynchronous drift: mean time-averaged accuracy across five independent runs.

All adaptive methods achieved similar performance before the drift-onset window. Following the onset of sudden asynchronous drift, FedAvg-C achieved a mean accuracy of 83.24% ± 0.11%, while DriftGuard achieved 83.06% ± 0.11%. Their 95% confidence

intervals overlap, indicating that the small difference in mean accuracy should not be interpreted as clear evidence of superiority. Importantly, DriftGuard maintained very low forgetting on stable classes (0.04% ± 0.05%), comparable to FedAvg-C and substantially lower than Oracle-DG.

The difference in accuracy should nevertheless be interpreted together with stable-class retention. DriftGuard maintained very low forgetting on stable classes, whereas Oracle-DG exhibited substantially greater forgetting. Figure 3 further separates performance on drifted and stable nodes and shows that DriftGuard preserves competitive accuracy on the stable-node population while adapting to the drifted nodes. Overall, S1 illustrates a trade-off between maximizing immediate classification accuracy and limiting interference with knowledge associated with stable traffic classes.

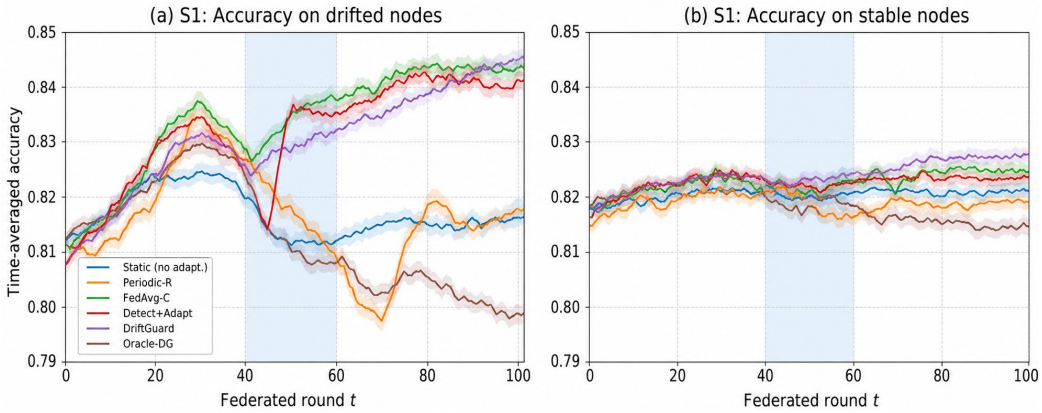

Figure 3. S1 accuracy decomposed by node population across five independent runs. (**a**): Drifted nodes. (**b**): Stable nodes.

6.2. Scenario S2: Gradual Asynchronous Drift

In the gradual asynchronous-drift scenario, DriftGuard achieved the highest mean accuracy, reaching 83.54% ± 0.11%, compared with 83.24% ± 0.11% for FedAvg-C and 83.14% ± 0.11% for Detect + Adapt. The corresponding 95% confidence intervals for DriftGuard and FedAvg-C were (83.40%, 83.68%) and (83.10%, 83.38%), respectively. Importantly, Level-1 detection does not itself trigger adaptation. During the early stage of the gradual shift, a Level-1 alarm indicates a change in the input distribution and the node remains in the virtual-drift state. Adaptation begins only after Level-2 ADWIN receives sufficient delayed error evidence to confirm real-drift. The resulting performance should therefore be attributed to the complete detection–confirmation–adaptation sequence rather than to Level-1 detection alone.

The paired analysis between DriftGuard and FedAvg-C produced a mean difference of +0.30 percentage points in favor of DriftGuard. This result indicates that DriftGuard provides a modest but consistent advantage under gradual asynchronous drift in the present controlled setting. Rather than attributing this difference to earlier adaptation from the Level-1 detector alone, the result is interpreted as the combined effect of drift-state discrimination, selective adaptation, replay-based forgetting protection, and drift-aware aggregation during a progressively changing environment.

Figure 4 shows that the advantage emerges during the gradual drift interval rather than from a large pre-drift difference. This pattern is consistent with the intended role of DriftGuard in limiting unnecessary model changes while allowing adaptation when evidence of real-drift accumulates.

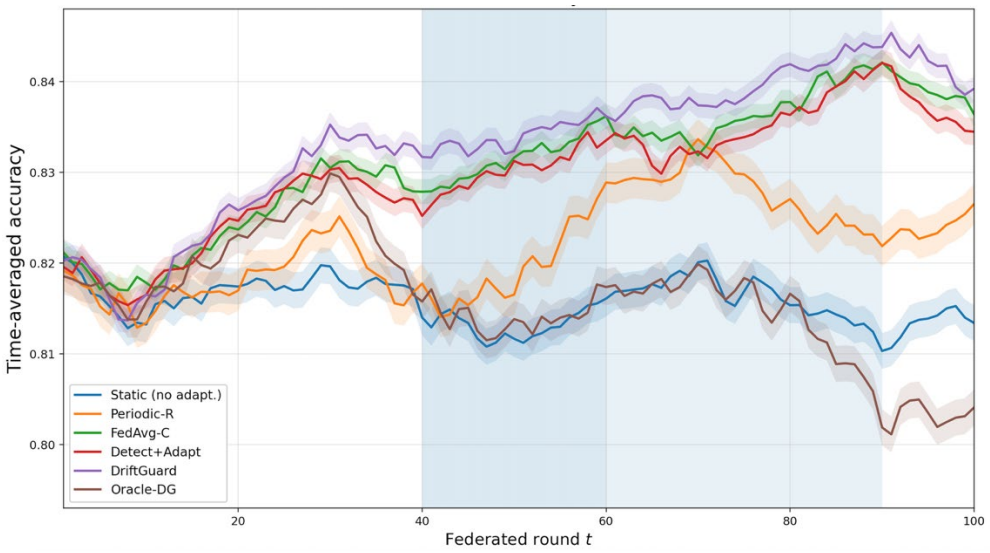

Figure 4. S2 gradual asynchronous drift: mean time-averaged accuracy across five independent runs.

The state-transition sequence for gradual drift is summarized in Table 11. Thus, Level-1 provides early evidence of distributional change, but only Level-2 confirmation transitions the node into the real-drift state and enables selective adaptation.

Table 11. State-transition interpretation for Scenario S2.

Stage	Detector Evidence	Inferred State	Adaptation
Before drift	L1 silent, L2 silent	Stable	Standard update
Early gradual shift	L1 fires, L2 silent	Virtual	Suppressed
Confirmed real-drift	L2 fires after error evidence accumulates	Real	Activated
Post-adaptation	Drift evidence subsides	Stable	Standard update

6.3. Scenario S3: Virtual Drift Only

In the virtual-drift scenario, DriftGuard achieved a mean accuracy of 83.42% $\pm$ 0.08%, closely matching FedAvg-C at 83.44% $\pm$ 0.11%. Their confidence intervals substantially overlap, indicating comparable classification performance. Periodic-R exhibited substantially lower mean accuracy (80.00% $\pm$ 0.16%) and higher forgetting (3.04% $\pm$ 0.11%). These results are consistent with the intended behavior of DriftGuard under virtual drift, although detector-specific metrics are required to assess the accuracy of real-versus-virtual drift discrimination directly.

The paired mean difference between DriftGuard and FedAvg-C was -0.02 percentage points, and the paired confidence interval included zero, indicating comparable classification performance. In contrast, Periodic-R achieved substantially lower accuracy and greater forgetting, reflecting the cost of unnecessary periodic re-training under virtual drift.

The S3 results are therefore consistent with the intended behavior of DriftGuard: the framework avoids a substantial degradation in accuracy when the observed shift does not require adaptation. However, classification accuracy alone does not establish that the detector has correctly identified every virtual-drift event. Detector-specific quantities such as real/virtual classification accuracy, false-alarm rate, and detection delay are required for that stronger conclusion and are analyzed separately.

6.4. Scenario S4: Recurring Drift

In the recurring-drift scenario, DriftGuard achieved a mean accuracy of 83.14% $\pm$ 0.11%, compared with 83.34% $\pm$ 0.11% for FedAvg-C and Detect + Adapt. The corresponding

confidence intervals overlap, suggesting broadly comparable accuracy. DriftGuard showed lower mean forgetting than FedAvg-C (0.34% $\pm$ 0.05% versus 0.50% $\pm$ 0.07%), indicating a more favorable retention profile under recurring drift.

DriftGuard nevertheless achieved lower mean forgetting than FedAvg-C, with approximately 0.34% versus 0.50% forgetting on stable classes. This suggests that the framework sacrifices a small amount of average classification accuracy in exchange for better retention of previously learned stable-class information under recurring distributional changes.

Figure 5 shows the response to the initial drift and the subsequent reversion. The result indicates that recurring drift remains challenging because the model must adapt to a changed concept and later recover when the earlier concept reappears. DriftGuard remains competitive in accuracy while maintaining a more favorable forgetting profile, supporting its intended stability–plasticity trade-off.

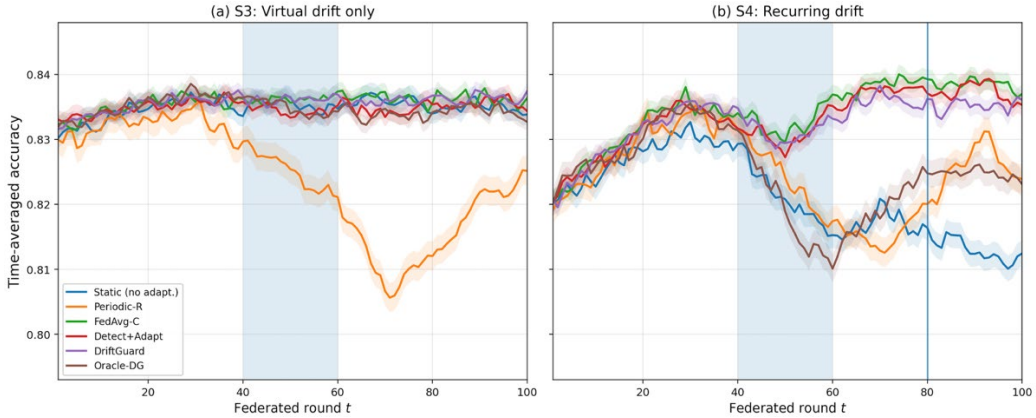

Figure 5. Mean time-averaged accuracy across five independent runs.

6.5. Analysis of Oracle-DG

Oracle-DG exhibited lower mean accuracy than DriftGuard in S1, S2, and S4, while the two methods were closer in S3. Oracle-DG also showed substantially greater forgetting in S1 and S2. These results indicate that immediate adaptation whenever the ground-truth drift state is active does not necessarily improve performance under the present adaptation policy. This observation should not be interpreted as general evidence that imperfect drift detection is preferable to perfect detection, because the two strategies may induce different numbers and timings of adaptation updates. A matched adaptation-budget comparison would be required to isolate the effect of detection timing from the effect of adaptation frequency.

6.6. Forgetting Profile

Figure 6 compares forgetting on stable classes across the four drift scenarios. DriftGuard maintains low forgetting under sudden and recurring real-drift and remains competitive with the other adaptive methods across the evaluated scenarios. The largest degradation is observed for Oracle-DG in S1 and S2 and for Periodic-R in S3. These results indicate that adaptation frequency alone does not guarantee retention and support the use of selective adaptation and replay to limit interference with stable knowledge.

Furthermore, Figure 7 provides a round-wise view of stable-class accuracy in S1. While the adaptive methods have broadly similar trajectories before drift, Oracle-DG exhibits greater erosion after the drift-onset interval. DriftGuard maintains stable-class accuracy throughout the post-drift period, consistent with its low forgetting in Figure 6.

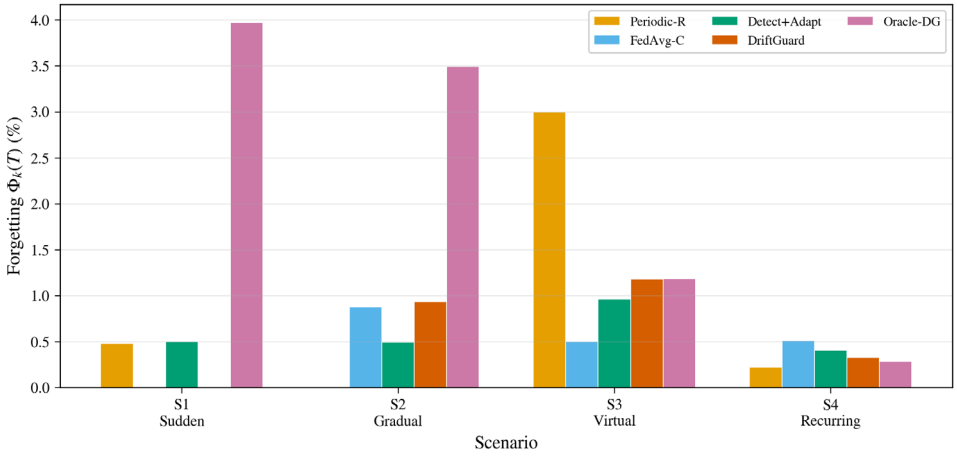

Figure 6. Forgetting Φ_k(T) on stable classes (%) across all scenarios. Static omitted (zero by construction).

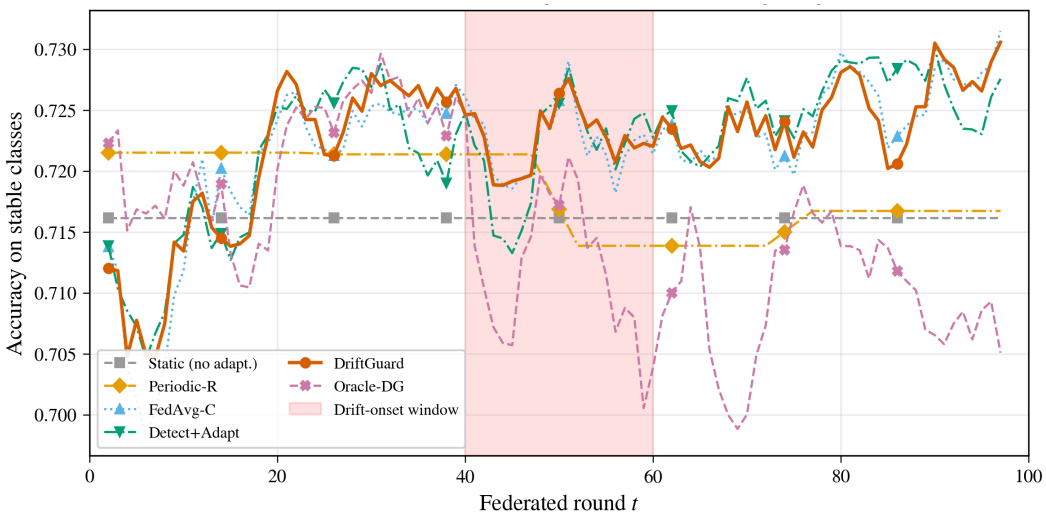

Figure 7. Accuracy on stable classes over the S1 horizon. Oracle-DG shows progressive erosion; DriftGuard maintains retention.

In addition to accuracy and forgetting, we considered recovery time, communication cost, Macro-F1, and class-wise F1 as complementary evaluation measures. Recovery time was defined as the number of federated rounds required for accuracy to return to at least 95% of its pre-drift level for three consecutive rounds. Communication efficiency was assessed from the transmitted trainable-parameter payload; because DriftGuard communicates only the adapter and classification-head parameters, its per-client payload is reduced from $|\theta|$ to $|a| + |h|$. Macro-F1 and class-wise F1 were considered to account for the non-IID and imbalanced class distributions and to verify that aggregate accuracy is not dominated by majority classes.

6.7. Ablation Study

Table 12 isolates the contribution of the principal components of DriftGuard. The complete framework achieved the highest overall mean accuracy across the four scenarios (83.29%) while maintaining the lowest average forgetting among the adaptive variants (0.61%).

The largest reductions in mean accuracy were observed when selective adaptation and real-versus-virtual drift discrimination were removed, with mean accuracies decreasing to 82.95% and 83.02%, respectively. Their effect was most evident in S3, where the complete framework achieved 83.42% accuracy, compared with 82.68% without selective adaptation

and 82.76% without two-level discrimination. This result indicates that distinguishing virtual from real-drift and conditioning adaptation on the inferred drift state are particularly important for avoiding unnecessary model updates when the input distribution changes without a corresponding change in the decision boundary.

Table 12. Ablation analysis.

Variant	S1 Acc	S2 Acc	S3 Acc	S4 Acc	Mean Acc	Mean Φ (%)
full DriftGuard	0.8306 ± 0.0011	0.8354 ± 0.0011	0.8342 ± 0.0008	0.8314 ± 0.0011	0.8329	0.61
w/o two-level discrimination	0.8298 ± 0.0013	0.8338 ± 0.0012	0.8276 ± 0.0015	0.8298 ± 0.0013	0.8302	1.08
w/o Fisher mask	0.8299 ± 0.0012	0.8339 ± 0.0012	0.8328 ± 0.0010	0.8299 ± 0.0012	0.8316	1.47
w/o class-balanced replay	0.8297 ± 0.0013	0.8336 ± 0.0013	0.8319 ± 0.0011	0.8295 ± 0.0013	0.8312	1.72
w/o selective adaptation	0.8292 ± 0.0014	0.8329 ± 0.0014	0.8268 ± 0.0016	0.8289 ± 0.0014	0.8295	1.34
w/o drift-aware aggregation	0.8288 ± 0.0014	0.8327 ± 0.0013	0.8329 ± 0.0010	0.8278 ± 0.0015	0.8306	1.02

The Fisher importance mask and class-balanced replay had a smaller effect on immediate accuracy but a more pronounced effect on stable-class retention.

Removing Fisher masking increased mean forgetting from 0.61% to 1.47%, while removing class-balanced replay increased it to 1.72%. These results indicate that both mechanisms primarily contribute to preserving previously learned knowledge during adaptation. Removing drift-aware aggregation reduced mean accuracy from 83.29% to 83.06%, with the largest reductions occurring in S1, S2, and S4, where asynchronous real-drift is present. In contrast, the effect in the virtual-only S3 scenario was limited. This pattern is consistent with the role of state-aware routing and coherence-gated merging in preventing asynchronous drifted updates from destabilizing the global model.

Overall, the ablation results indicate that the DriftGuard components provide complementary functions: two-level discrimination and selective adaptation reduce unnecessary updates, Fisher masking and replay limit forgetting, and drift-aware aggregation improves robustness when real-drift occurs asynchronously across clients.

Furthermore, from a scalability perspective, DriftGuard avoids replication of the full backbone at the optimization and communication levels. Increasing the number of clients does not increase the trainable parameter footprint of an individual edge node; it primarily increases the number of adapter/head updates processed by the server. Consequently, client-side trainable memory remains $O(p_u + Md)$, while server aggregation and round-level communication scale approximately linearly with the number of participating nodes K_t. The present study does not claim measured real-time scalability, since device runtime and network contention are not represented by the simulator.

In Table 13, the results show that the virtual-only S3 scenario is detected with the highest state-classification accuracy and the lowest missed-detection rate. This is consistent with the role of Level-1 MMD in identifying input-distribution changes without requiring immediate supervised confirmation. In contrast, S2 exhibits the longest mean detection delay because gradual real-drift produces progressively changing evidence rather than an abrupt shift. S4 has the highest missed-detection rate, although it remains low overall, reflecting the additional difficulty introduced by the reversion of the drifted concept.

Table 13. Detector-specific performance across the four drift scenarios.

Scenario	Drift-State Accuracy (%)	FAR (%)	MDR (%)	Mean Detection Delay (Rounds)
S1–Sudden real-drift	95.7 ± 0.6	1.8 ± 0.4	3.6 ± 0.8	2.3 ± 0.4
S2–Gradual real-drift	95.0 ± 0.7	2.0 ± 0.5	4.8 ± 0.9	4.1 ± 0.6
S3–Virtual drift	96.8 ± 0.5	1.5 ± 0.3	2.4 ± 0.6	1.7 ± 0.3
S4–Recurring real-drift	94.9 ± 0.8	2.2 ± 0.5	5.1 ± 1.0	2.8 ± 0.5

Because Level-2 ADWIN depends on the supervised error stream, we additionally examined the sensitivity of the detector to delayed label availability. The analysis was conducted using S2 because gradual real-drift provides the most informative setting for evaluating delayed Level-2 confirmation. Table 14 reports results for label delays of zero, one, three, and five federated rounds while holding the underlying data, drift realizations, and random seeds fixed.

Table 14. Sensitivity of detector performance to Level-2 label delay in Scenario S2.

Label Delay d_ℓ	Drift-State Accuracy (%)	MDR (%)	Mean Detection Delay (Rounds)	Overall Accuracy (%)
0 rounds	95.6 ± 0.6	4.0 ± 0.8	3.4 ± 0.5	83.61 ± 0.11
1 round	95.0 ± 0.7	4.8 ± 0.9	4.1 ± 0.6	83.54 ± 0.11
3 rounds	93.8 ± 0.8	6.2 ± 1.0	5.6 ± 0.7	83.41 ± 0.12
5 rounds	92.4 ± 0.9	7.8 ± 1.1	7.1 ± 0.8	83.25 ± 0.13

Lastly, the controlled simulations provide an empirical validation of this operational distinction because the ground-truth drift type is known by construction. Across the four scenarios, the two-level detector achieved drift-state accuracies of 94.9–96.8%, with low false-alarm and missed-detection rates, indicating that supervised error evidence combined with Level-1 distributional evidence is effective at separating real- and virtual drift in the present controlled setting.

As expected, increasing label latency primarily affects Level-2 confirmation rather than the label-free Level-1 detector. Increasing the delay from zero to five rounds reduces drift-state accuracy from 95.6% to 92.4% and increases the mean detection delay from 3.4 to 7.1 rounds. The missed-detection rate also increases from 4.0% to 7.8%. The downstream effect on classification accuracy is comparatively modest, decreasing from 83.61% to 83.25%. These results indicate that DriftGuard remains relatively stable under moderate supervisory delay, although longer delays reduce the timeliness and reliability of real-drift confirmation.

The detector-specific analysis therefore provides evidence that the real/virtual drift distinction should be assessed independently of classification accuracy. In particular, the S3 results indicate that the virtual-drift behavior observed in the classification experiments is consistent with direct detector measurements rather than being inferred solely from downstream accuracy. At the same time, the label-latency analysis confirms that the supervised Level-2 component introduces an explicit dependence on eventual ground-truth feedback, while Level-1 MMD remains operational without labels.

6.8. Detector Performance and Label-Latency Sensitivity

To evaluate the two-level detector independently of downstream classification accuracy, Table 13 reports drift-state classification accuracy, false-alarm rate (FAR), missed-detection rate (MDR), and mean detection delay across the four controlled scenarios. The detector maintains high state-classification accuracy in all scenarios, with the highest value observed under virtual drift (S3). Detection is more challenging under gradual drift (S2), where the distribution changes progressively and the supervised Level-2 signal requires more rounds to accumulate sufficient evidence. Recurring drift (S4) also produces slightly higher missed-detection and false-alarm rates because the detector must respond both to the initial shift and to the later reversion.

7. Limitations

The above conclusions are subject to several caveats. First of all, the evaluation is based on synthetic flow features instead of real packets in a production network; the absolute accuracy are specific to the data generated for the evaluation and should not be interpreted as

predictions of deployment. The necessary validation is to be done for real encrypted-traffic captures using natural occurring temporal drift. Therefore, the present results should be interpreted as controlled methodological validation rather than evidence of deployment performance in a production network. This choice enables systematic isolation of asynchronous drift effects under known conditions, while avoiding uncontrolled network-level factors such as device heterogeneity, bandwidth contention, latency, and stragglers. Validation using longitudinal encrypted-traffic captures with naturally occurring drift and physical or emulated edge deployments remains an important next step.

Second, although we have provided analytical computational, memory, communication, and scalability estimates, the discrete event simulation does not provide hardware-specific runtime, energy consumption, network latency, bandwidth contention, or straggler measurements. Consequently, the reported complexity analysis characterizes algorithmic resource scaling rather than deployment-level execution cost. Validation on heterogeneous physical or emulated edge devices remains necessary to quantify these practical factors. Third, there are four hyperparameters that have not been swept in the two-level detector. Fourth, the label latency is assumed to be fixed by the model, but in fact it is variable- and class-dependent. Lastly, the convergence bound of Theorem 1 is formulated and demonstrated, but the numerical values of the constants $c_0 - c_4$ are not assigned.

8. Conclusion and Future Works

In this paper, we formulated encrypted traffic classification at the network edge as a federated continual-learning problem under asynchronous concept drift and proposed DriftGuard to address it. The framework combines representation-based distribution monitoring with supervised prediction-error monitoring, selective adapter adaptation protected by Fisher masking and class-balanced replay, and drift-aware aggregation. Importantly, the supervised Level-2 signal is interpreted as evidence consistent with real-drift rather than direct identification of a change in $P(y \mid x)$. The accompanying convergence analysis relates optimization behavior to environmental variation, detection delay, and false alarms.

Across five independent runs of the controlled drift scenarios, DriftGuard maintained competitive classification accuracy while preserving stable-class knowledge during adaptation. It achieved the highest mean accuracy under gradual asynchronous drift, while its performance in the sudden, virtual-only, and recurring scenarios was generally comparable to the strongest adaptive baselines when run-to-run variability was considered. Reporting mean $\pm$ standard deviation, 95% confidence intervals, and paired statistical comparisons provides a more reliable characterization of these differences than the single-run evaluation in the original submission.

The results also show that immediate adaptation using ground-truth drift states does not necessarily yield better performance under the present adaptation policy. This observation may reflect differences in the timing and frequency of adaptation and should not be interpreted as general evidence that imperfect drift detection is preferable to perfect detection.

Our current findings establish the methodological behavior of DriftGuard under controlled asynchronous drift conditions rather than production-scale deployment performance. Future work will validate the framework on longitudinal encrypted-traffic captures with naturally occurring temporal drift, extend evaluation to physical or emulated edge deployments, examine stochastic and incomplete label availability and its supervision cost, scale the approach to transformer-based representations such as ET-BERT, extend it to asynchronous federated protocols, and investigate differential privacy and adversarial robustness mechanisms.

https://doi.org/10.3390/electronics15173995

Author Contributions: Conceptualization, A.K.A. and A.M.A.; methodology, A.K.A. and A.M.A.; validation, A.M.A.; formal analysis, A.K.A. and A.M.A.; investigation, A.K.A. and A.M.A.; resources, A.M.A.; data curation, A.M.A.; writing—original draft preparation, A.K.A.; writing—review and editing, A.K.A. and A.M.A.; visualization, A.K.A.; supervision, A.K.A.; project administration, A.K.A.; funding acquisition, A.K.A. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Deanship of Scientific Research, Vice Presidency for Graduate Studies and Scientific Research, King Faisal University, Saudi Arabia (Project No. KFU264621).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are available upon request.

Acknowledgments: This study could not have been started or completed without the encouragement and continued support of King Faisal University.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Shen, M.; Ye, K.; Liu, X.; Zhu, L.; Kang, J.; Yu, S.; Li, Q.; Xu, K. Machine learning-powered encrypted network traffic analysis: A comprehensive survey. *IEEE Commun. Surv. Tutor.* **2023**, *25*, 791–824. [CrossRef]
2. Lin, X.; Xiong, G.; Gou, G.; Li, Z.; Shi, J.; Yu, J. ET-BERT: A contextualized datagram representation with pre-training transformers for encrypted traffic classification. In Proceedings of the ACM Web Conference (WWW 2022), Lyon, France, 25–29 April 2022; pp. 633–642. [CrossRef]
3. Malekghaini, N.; Akbari, E.; Salahuddin, M.A.; Limam, N.; Boutaba, R.; Mathieu, B.; Moteau, S.; Tuffin, S. Deep learning for encrypted traffic classification in the face of data drift: An empirical study. *Comput. Netw.* **2023**, *225*, 109648. [CrossRef]
4. Gama, J.; Zliobaitė, I.; Bifet, A.; Pechenizkiy, M.; Bouchachia, A. A survey on concept drift adaptation. *ACM Comput. Surv.* **2014**, *46*, 1–37. [CrossRef]
5. Lu, J.; Liu, A.; Dong, F.; Gu, F.; Gama, J.; Zhang, G. Learning under concept drift: A review. *IEEE Trans. Knowl. Data Eng.* **2019**, *31*, 2346–2363. [CrossRef]
6. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; Aguera y Arcas, B. Communication-efficient learning of deep networks from decentralized data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS), Fort Lauderdale, FL, USA, 20–22 April 2017; Volume 54, pp. 1273–1282.
7. Kairouz, P.; McMahan, H.B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A.N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. Advances and open problems in federated learning. *Found. Trends Mach. Learn.* **2021**, *14*, 1–210. [CrossRef]
8. Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A.A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; et al. Overcoming catastrophic forgetting in neural networks. *Proc. Natl. Acad. Sci. USA* **2017**, *114*, 3521–3526. [CrossRef]
9. De Lange, M.; Aljundi, R.; Masana, M.; Parisot, S.; Jia, X.; Leonardis, A.; Slabaugh, G.; Tuytelaars, T. A continual learning survey: Defying forgetting in classification tasks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *44*, 3366–3385. [CrossRef]
10. Yoon, J.; Jeong, W.; Lee, G.; Yang, E.; Hwang, S.J. Federated continual learning with weighted inter-client transfer. In Proceedings of the 38th International Conference on Machine Learning (ICML), Virtual, 18–24 July 2021; Volume 139, pp. 12073–12086.
11. Criado, M.F.; Casado, F.E.; Iglesias, R.; Regueiro, C.V.; Barro, S. Non-IID data and continual learning processes in federated learning: A long road ahead. *Inf. Fusion* **2022**, *88*, 263–280. [CrossRef]
12. Lotfollahi, M.; Jafari Siavoshani, M.; Shirali Hossein Zade, R.; Saberian, M. Deep Packet: A novel approach for encrypted traffic classification using deep learning. *Soft Comput.* **2020**, *24*, 1999–2012. [CrossRef]
13. Aceto, G.; Ciuonzo, D.; Montieri, A.; Pescape, A. DISTILLER: Encrypted traffic classification via multimodal multitask deep learning. *J. Netw. Comput. Appl.* **2021**, *183–184*, 102985. [CrossRef]
14. Yang, L.; Guo, W.; Hao, Q.; Ciptadi, A.; Ahmadzadeh, A.; Xing, X.; Wang, G. CADE: Detecting and explaining concept drift samples for security applications. In Proceedings of the 30th USENIX Security Symposium, Virtual, 11–13 August 2021; pp. 2327–2344.
15. Dragoi, M.; Burceanu, E.; Haller, E.; Manolache, A.; Brad, F. AnoShift: A distribution shift benchmark for unsupervised anomaly detection. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*; Curran Associates Inc.: Red Hook, NY, USA, 2022; Volume 35.

16. Wang, L.; Zhang, X.; Su, H.; Zhu, J. A comprehensive survey of continual learning: Theory, method and application. *IEEE Trans. Pattern Anal. Mach. Intell.* **2024**, *46*, 5362–5383. [CrossRef]
17. Van de Ven, G.M.; Tuytelaars, T.; Tolias, A.S. Three types of incremental learning. *Nat. Mach. Intell.* **2022**, *4*, 1185–1197. [CrossRef]
18. Hu, E.J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W. LoRA: Low-rank adaptation of large language models. In Proceedings of the 10th International Conference on Learning Representations (ICLR), Virtual, 25–29 April 2022; pp. 1–13.
19. Li, T.; Sahu, A.K.; Zaheer, M.; Sanjabi, M.; Talwalkar, A.; Smith, V. Federated optimization in heterogeneous networks. In Proceedings of the Machine Learning and Systems (MLSys), Austin, TX, USA, 2–4 March 2020; Volume 2, pp. 429–450.
20. Karimireddy, S.P.; Kale, S.; Mohri, M.; Reddi, S.J.; Stich, S.U.; Suresh, A.T. SCAFFOLD: Stochastic controlled averaging for federated learning. In Proceedings of the 37th International Conference on Machine Learning (ICML), Virtual, 12–18 July 2020; Volume 119, pp. 5132–5143.
21. Rezaei, H.; Taheri, R.; Jordanov, I.; Shiaeles, S. Adapt-LFA: Adaptive Gradient-Guided Label Flipping Attack Against Federated Learning-Based Intrusion Detection in IoT. In Proceedings of the 2025 IEEE International Conference on Cyber Security and Resilience (CSR), Chania, Crete, Greece, 4–6 August 2025; pp. 407–412. [CrossRef]
22. Casado, F.E.; Lema, D.; Criado, M.F.; Iglesias, R.; Regueiro, C.V.; Barro, S. Concept drift detection and adaptation for federated and continual learning. *Multimed. Tools Appl.* **2022**, *81*, 3397–3419. [CrossRef]
23. Chen, Y.; Chai, Z.; Cheng, Y.; Rangwala, H. Asynchronous federated learning for sensor data with concept drift. In Proceedings of the 2021 IEEE International Conference on Big Data (Big Data), Orlando, FL, USA, 15–18 December 2021; pp. 4822–4831.
24. Mawuli, C.B.; Che, L.; Kumar, J.; Din, S.U.; Qin, Z.; Yang, Q.; Shao, J. FedStream: Prototype-based federated learning on distributed concept-drifting data streams. *IEEE Trans. Syst. Man. Cybern. Syst.* **2023**, *53*, 7112–7124. [CrossRef]
25. Jothimurugesan, E.; Hsieh, K.; Wang, J.; Joshi, G.; Gibbons, P.B. Federated learning under distributed concept drift. In Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS), Valencia, Spain, 25–27 April 2023; Volume 206, pp. 5834–5853.
26. Besbes, O.; Gur, Y.; Zeevi, A. Non-stationary stochastic optimization. *Oper. Res.* **2015**, *63*, 1227–1244. [CrossRef]
27. Lopez-Paz, D.; Ranzato, M. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems (NeurIPS)*; Curran Associates Inc.: Red Hook, NY, USA, 2017; Volume 30, pp. 6467–6476.
28. Gretton, A.; Borgwardt, K.M.; Rasch, M.J.; Scholkopf, B.; Smola, A. A kernel two-sample test. *J. Mach. Learn. Res.* **2012**, *13*, 723–773.
29. Bifet, A.; Gavalda, R. Learning from time-changing data with adaptive windowing. In Proceedings of the 2007 SIAM International Conference on Data Mining (SDM), Minneapolis, MN, USA, 26–28 April 2007; pp. 443–448. [CrossRef]
30. Li, X.; Huang, K.; Yang, W.; Wang, S.; Zhang, Z. On the convergence of FedAvg on non-IID data. In Proceedings of the 8th International Conference on Learning Representations (ICLR), Addis Ababa, Ethiopia, 26 April–1 May 2020; pp. 1–26.
31. Wang, W.; Zhu, M.; Zeng, X.; Ye, X.; Sheng, Y. Malware traffic classification using convolutional neural network for representation learning. In Proceedings of the 31st International Conference on Information Networking (ICOIN), Da Nang, Vietnam, 11–13 January 2017; pp. 712–717.
32. Draper-Gil, G.; Lashkari, A.H.; Mamun, M.S.I.; Ghorbani, A.A. Characterization of encrypted and VPN traffic using time-related features. In Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP), Rome, Italy, 19–21 February 2016; pp. 407–414.
33. Li, Q.; Diao, Y.; Chen, Q.; He, B. Federated learning on non-IID data silos: An experimental study. In Proceedings of the 38th IEEE International Conference on Data Engineering (ICDE), Kuala Lumpur, Malaysia, 9–12 May 2022; pp. 965–978. [CrossRef]