Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Domain-aware federated log anomaly detection: mapping failure modes in heterogeneous and evolving domains

vansh MMundhra
Vellore Institute of Technology University
Sankari M
sankari.m@vit.ac.in
Vellore Institute of Technology University
Pinak Debnath
Vellore Institute of Technology University

Article

Keywords: Federated learning, log anomaly detection, data heterogeneity, domain skew, catastrophic forgetting, continual learning, DeepLog, model aggregation

Posted Date: November 20th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11783752/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Domain-Aware Federated Log Anomaly Detection: Mapping Failure Modes in Heterogeneous and Evolving Domains

Vansh Mundhra[1], Sankari M[1*], Pinak Debnath[1]

[1]School of Computer Science and Engineering, Vellore Institute of Technology, Chennai, India.

*Corresponding author(s). E-mail(s): sankari.m@vit.ac.in;
Contributing authors: vansh.mundhra2023@vitstudent.ac.in;
pinak.debnath2023@vitstudent.ac.in;

Abstract

Federated log anomaly detection must accommodate heterogeneous log domains and evolving client populations, yet the robustness of lightweight statistical detectors and the bounded size of their models have been studied mainly in homogeneous settings. We investigate these properties under cross-domain heterogeneity and sequential domain arrival. Experiments on HDFS and BGL, unified through a 427-event vocabulary, yield three findings. First, domain-blind range-based aggregation fails on a tightly distributed domain, reducing HDFS Length F1 from 0.538 to 0.000, whereas domain-aware aggregation restores it to 0.561 ± 0.024 across five seeds. Second, simultaneous mixed Federated Averaging does not degrade deep models, including those with learned embeddings. Third, sequential domain arrival causes catastrophic forgetting when domain vocabularies share a feature space, reducing HDFS F1 from 0.70 to 0.04 (forgetting 0.665 ± 0.003, three seeds). We formalize range collapse as a containment condition and validate its predictions in three- and four-domain federations including OpenStack and Hadoop. Transformers also forget, but a replay buffer of about 10% substantially mitigates this; a memory-free elastic weight consolidation alternative is also evaluated. Finally, we present a deployable auditor that predicts silent failure modes from summary statistics and recommends corrections before training. This yields a practical diagnostic framework.

Keywords: Federated learning, log anomaly detection, data heterogeneity, domain skew, catastrophic forgetting, continual learning, DeepLog, model aggregation

I. Introduction

System logs are the main forensic record of large software and infrastructure deployments, and spotting anomalous log sequences is central to reliability, availability, and security engineering [1]. Logs increasingly originate from distributed, privacy-sensitive, or bandwidth-constrained sources (Internet of Things (IoT) fleets, industrial networks, smart grids, multi-tenant clouds), which often makes centralizing them for analysis infeasible. Federated learning [2] is one way around this: clients train locally and send only model updates, so the raw logs never leave their premises.

Pustozerova *et al.* [3] recently benchmarked *lightweight statistical* detectors (sequence-length ranges, sets of seen events, count-vector nearest neighbours) against *deep* detectors (DeepLog [4], LogAnomaly [5]) under Federated Averaging (FedAvg). Their study is careful, and it reports two encouraging properties. (C1) Lightweight detectors are approximately *invariant to client data distribution*, so their federated F_1 matches the centralized value even under non-IID splits. (C2) Their model size is *bounded by a finite event vocabulary*, so memory stabilizes once the vocabulary saturates.

The gap. That evaluation is *homogeneous*. Its non-IID setting is *quantity skew*: each client receives a different *amount* of training data, drawn from a log-normal allocation, but every client still holds a slice of the *same* dataset (HDFS *or* BGL, never both). Real federations differ in two ways. They are *domain-heterogeneous*, since different services, systems, and vendors produce categorically different logs, and they are *evolving*, since new log sources keep being onboarded. Neither claim had been tested under those conditions; the public artifact of [3] ships hdfs_* and bgl_* configurations but no mixed one. This matters in practice. Under mixing We find that detection can collapse *without any error or crash*, losing up to 99.97% of a detector's anomaly coverage (Section XIII). An operator would never see that blind spot, and a few bytes of the right safeguard would have prevented it.

We place HDFS and BGL clients in *one* federation over a merged event vocabulary and, in a separate experiment, let the domains arrive *sequentially*; C1 and C2 are tested directly in both. The findings form a compact map of when federated log anomaly detection is safe, when it is not, and why. In brief:

- **H1 (range aggregation).** Domain-blind range aggregation *collapses* on the tighter-distributed domain. BGL's large length spread widens the shared global range until almost no HDFS anomaly is flagged, which drives HDFS Length F_1 from 0.538 to 0.000. A simple *domain-aware* aggregation (keep per-domain range parameters and route by a domain tag) restores HDFS to 0.561, its single-domain level, without touching BGL. This is a clean counterexample to C1 under domain skew.
- **H2 (simultaneous mixing).** FedAvg on DeepLog does *not* degrade when HDFS and BGL clients train together, and BGL even improves. The result holds under both a scalar-identifier input and a learned embedding.
- **H3 (sequential arrival) and the mechanism.** When the new domain arrives *sequentially* instead, the deep model *catastrophically forgets* the earlier one (F_1 $0.70 \rightarrow 0.04$), but only if the input representation forces the two vocabularies to share a feature space. A controlled scalar-versus-embedding experiment points to the

representation, not the aggregation method, as the cause. Set-union detectors never forget but grow $\times 3.7$ when a domain is added, which breaks C2 under evolution.

The contributions of the proposed system are as follows. (i) A merged cross-domain vocabulary and dataloader that let heterogeneous log clients share one federation; prior pipelines had no such artifact. (ii) A demonstration, across five seeds and both independent and identically distributed (IID) and non-IID splits, that domain-blind range aggregation fails under domain skew. We *prove* the failure as a containment condition (Proposition 1) and give a minimal domain-aware fix that restores single-domain performance and routes at inference with no oracle tag (Corollary 1; 100% accuracy). (iii) A controlled 2×2 map showing that *simultaneous* mixing is safe while *sequential* arrival is catastrophic *if and only if* the representation shares a feature space. The forgetting is *directional* and *architecture-general* (a long short-term memory (LSTM) network and a self-attention Transformer both forget). We evaluate two mitigations, replay and EWC, each of which removes it, and compare against the episodic-memory method Averaged Gradient Episodic Memory (A-GEM), with which simple replay is Pareto-competitive. (iv) An extension beyond the pair to *three- and four-domain* federations (adding OpenStack and Hadoop). It reproduces the range and routing results exactly and shows that the containment condition is *relative*: a broader domain demotes the previous widener to a victim. (v) A deployable *safety auditor* that, from per-client summary statistics alone (event-id blocks and length ranges, never raw logs), predicts which failures a planned federation will hit and prescribes the fix. It is correct on 14 of 15 measured configurations. (vi) A CPU-only study in which every reported number comes from an actual run.

II. Literature Survey

This work sits at the intersection of four research strands: log-based anomaly detection, federated anomaly detection under heterogeneity, federated log analytics, and continual learning (including its federated and log-specific forms). Following the scope of this survey, We review and compare only work from the last three years (2023–2026). The foundational techniques that We build on (the Drain parser, the DeepLog detector, FedAvg, EWC, and A-GEM) are cited where they are used (Sections V and IX–XI-A) rather than reviewed here. We take each strand in turn and then position the contribution against the closest recent work in Table I.

A. *Recent work on log anomaly detection*

Recent surveys agree that log anomaly detection has settled into two broad families of method: statistical detectors built on hand-crafted features such as event counts and length statistics, and deep detectors that model the event sequence directly. Landauer *et al.* [1] review the deep-learning branch and its open problems. A critical review of the commonly used datasets [6] finds that no single detector dominates across datasets, which is why practical systems combine several detectors into an ensemble. It is also why We reuse the Length, Known Events, and event count vector clustering (ECVC) ensemble structure, where ECVC scores a sequence by the L1 distance between its normalized event histogram and the nearest normal one [6]. The HDFS and BGL benchmarks used come from the Loghub collection [7]. Deployed systems also change over time. Tian *et al.* [8] tackle the resulting distribution shift (changing event types and frequencies) with an online meta-learning detector evaluated on HDFS and BGL.

All of this work, though, deals with a *single* log domain at a time, even when that domain evolves. The question here is a different one: what happens when the clients of one federation hold *different* log domains at once.

B. *Federated anomaly detection under heterogeneity*

Federated learning keeps raw data on the client and aggregates model updates at a server. Its central difficulty is that clients are rarely alike. A recent, wide-ranging review of federated anomaly detection [9] frames the problem around data heterogeneity (different distributions and feature spaces) and client heterogeneity (different resources), and observes that heterogeneity makes accurate and reliable detectors hard to build. The heterogeneity studied here, domain skew, is the case where clients hold data from different generative processes with largely disjoint vocabularies. It is an extreme form of feature skew, and it is also the common case in real log deployments. One of the results partly cuts against the usual story that heterogeneity hurts federated training: for a next-event LSTM with a scalar-identifier input, mixing domains simultaneously does *not* degrade accuracy, because the domains occupy disjoint input ranges and never compete for the same parameters. The proposed works show that this robustness depends on the representation, which reconciles the two views rather than contradicting either.

C. *Federated log anomaly detection*

The closest study to this one is Pustozerova *et al.* [3]. The proposed works adopt its lightweight methods, datasets, and FedAvg pipeline, and test its two claims (C1 distribution invariance, C2 bounded growth) outside the homogeneous setting it used. Himler *et al.* [10] study federated *deep* log detection and list cross-dataset heterogeneity among its open challenges. FedLAD [11] is a testbed for running federated log anomaly detection across different client configurations, and it is built to be modular and adaptive. FedXLog [12] is the most recent entry. It copes with heterogeneous logs through hierarchical feature extraction and hash feature alignment, and it adds explainability (through gradient-trajectory aggregation) and a lightweight distilled model. Its stated goals are accuracy, interpretability, and efficiency; characterizing when aggregation fails is not among them. These efforts either keep every client on the same dataset, so that the skew is in quantity or labels and never in domain, or they supply infrastructure or models that *could* handle varied clients without reporting a mixed-domain failure, its mechanism, or a fix. None of them tests the two published invariance and bounded-size claims in a *single federation that mixes distinct log domains*, under *sequential* domain arrival, or identifies the input representation as the variable that governs forgetting. That gap is the target of the proposed works.

D. *Continual and federated continual learning, including for logs*

Sequential domain arrival is a continual-learning problem, and its defining failure is catastrophic forgetting: a drop in performance on earlier tasks after training on a new one. Combining continual learning with federation makes this worse, because forgetting compounds with non-IID aggregation. Dupuy *et al.* [13] quantify the effect, and a recent survey [14] sorts the federated continual setting for edge AI into class-, domain-, and task-incremental regimes. The closest *security* analogue is FL-IIDS [15], a federated incremental intrusion detector that explicitly targets catastrophic forgetting, but on network-flow features with *class*-incremental tasks, not heterogeneous log

domains. In the log setting, CEDLog [16] adds elastic weight consolidation (EWC) [17] to a distributed detector so that it can be updated without forgetting, and OMLog [8] adapts online to shifting logs. Each proposes a defense. Neither studies a federation of heterogeneous domains, and neither states when forgetting must occur. The H3 result adds a concrete, reproducible instance in log analytics. It shows that whether forgetting occurs at all is decided by the input representation (a shared versus disjoint feature space), a lever that is orthogonal to the penalties and rehearsal buffers these works study.

E. *Positioning and novelty*

Table I compares this paper with the closest work of the last three years along the seven axes that define the contribution. No study in this window combines a domain-heterogeneous federation (clients holding categorically different log sources in one federation, not slices of one dataset) with evolving, sequential arrival. None pairs that setting with (a) a proven failure condition, (b) evidence that the mechanism is architecture-general, (c) a benchmarked family of mitigations, and (d) a validated, deployable pre-training auditor. Prior work either studies a different kind of heterogeneity (quantity or class skew rather than domain) or offers a fix without a predictive account of *when* the failure occurs. The novelty lies in the combination: a safety map of when federated log anomaly detection is safe; a containment proof that explains the lightweight collapse; the identification of a shared feature space as the single lever behind the deep-model forgetting, confirmed across an LSTM and a Transformer and across up to four domains; and a summary-statistics-only auditor that turns the map into a check an operator can run before training.

TABLE I

Positioning against the closest prior work from the last three years (2023–2026) along the seven axes that define the contribution. *Het. domains*: clients hold categorically different log/data sources in one federation (not slices of one dataset). *Evolving*: new sources or distributions arrive over time. *LW+Deep*: both lightweight and deep detectors studied. *Theory*: a proven failure/safety condition. *Mitig.*: a forgetting or collapse defense is evaluated. *Auditor*: a pre-deployment predictor of failures. ✓ = yes, ∼ = partial, — = no.

III. Problem Setting

A federation of K clients is considered, each holding semi-supervised training data (normal sequences only) drawn from one *domain* $d \in \mathrm{D}$. Each domain has its own set of event templates, and the global model is formed by aggregating the client models. Two published claims of [3] are tested:

- **C1 (distribution invariance).** For lightweight detectors built by deterministic set-union or min–max aggregation, the global model is "independent of client data distributions," so federated F_1 equals the centralized F_1.
- **C2 (bounded growth).** Model size is bounded by the finite event vocabulary; once the vocabulary saturates, memory stabilizes.

Work (year)	Federated	Het. domains	Evolving	LW+Deep	Theory	Mitig.	Auditor
Pustozerova *et al.* [3] (2026, fed. lightweight log)	✓	—	—	✓	—	—	—
Gu *et al.* [12] (2026, FedXLog)	✓	∼	—	—	—	—	—
Xiang *et al.* [9] (2026, fed. anomaly detection review)	✓	∼	—	—	—	—	—
FedLAD [11] (2025, testbed)	✓	∼	—	—	—	—	—
CEDLog [16] (2025, distributed, EWC)	—	—	✓	—	—	✓	—
Himler *et al.* [10] (2024, fed. deep log)	✓	∼	—	—	—	—	—
OMLog [8] (2024, online meta-learning)	—	—	✓	—	—	∼	—
FL-IIDS [15] (2024, fed. incremental IDS)	✓	—	✓	—	—	✓	—
FCL survey [14] (2024, continual, non-log)	✓	∼	✓	—	—	✓	—
This work	✓	✓	✓	✓	✓	✓	✓

Both claims were validated only under *quantity skew within one dataset*. They are evaluated here under *domain skew* (Sections VI and VII) and under *sequential domain arrival* (Section VIII).

IV. Method

A. *Merged cross-domain vocabulary*

To let heterogeneous clients share one federation, a single global event space is built. Each domain's parsed templates form a contiguous block of identifiers: HDFS takes global ids [0, 33) and BGL [33, 427), giving a merged vocabulary of 427 events. A domain tag travels with every sequence, which makes per-domain evaluation and domain-aware routing possible. Because HDFS and BGL both reuse small integers for *different* templates, each event is keyed as domain:id. HDFS "5" then maps to global id 0, while BGL "5" maps to global id 43, and the two never collide. This mirrors the reference pipeline's per-dataset global dictionary, extended across datasets.

B. *Federation construction*

Each domain contributes N_d clients. Following the reference convention, 1% of each domain's normal sequences form the training pool, and the remainder is held out for evaluation. The pool is split across the domain's clients either uniformly (IID) or with a log-normal quantity skew ($\sigma = 0.25$), as in the reference non-IID setting. Unless stated otherwise, the mixed federation has 3 HDFS + 2 BGL clients over the merged 427-event vocabulary.

C. *Detectors*

Length learns the [min, max] length of normal sequences and flags anything outside it; domain-blind FedAvg takes the global min-of-mins and max-of-maxes. *Known Events* learns the set of normal event ids (FedAvg is set union) and flags any unseen

event. *ECVC* scores a sequence by the L_1 distance between its normalized event-count histogram and the nearest normal histogram, and applies a supervised threshold. *DeepLog* [4] is a two-layer, 64-unit LSTM that predicts the next event from a window of length 10. A sequence counts as anomalous when the true next event is missing from the top-9 predictions, and FedAvg averages the client weights. To study representation effects, two DeepLog input encodings are supported: the reference's raw scalar event id (input size 1) and a learned 16-dimensional embedding.

D. *Domain-aware aggregation (the fix)*

Instead of one global range, domain-aware Length keeps a *per-domain* [min, max], aggregated only across clients of the same domain, and routes each test sequence to its domain's range using the domain tag. Clients of a domain still collaborate, but the cross-domain contamination behind H1 is gone. A domain's aggregated range is the min–max over the union of that domain's client data, so it is *invariant* to how the data is partitioned across clients, and the fix is robust to quantity skew by construction.

Removing the oracle tag. The routing above assumes each test sequence carries a domain tag, which a real deployment may not provide, so two tag-free strategies are considered. *Vocabulary-block routing* infers a sequence's domain from its event ids. Each domain owns a contiguous, disjoint id block ([0, 33) for HDFS, [33, 427) for BGL), so a sequence's domain is the majority block of its ids, and a sequence with an id outside every block is flagged as anomalous (unknown domain). *Conservative routing* skips inference entirely and flags a sequence only if its length is outside *every* domain's range. Both are evaluated against the oracle in Section VI. Vocabulary-block routing reproduces the oracle-tag F_1 exactly, so the fix needs no external domain label; conservative routing collapses on the tighter domain.

V. Experimental Setup

The HDFS and BGL benchmarks from Loghub [7] are used, parsed into event-id sequences with the Drain parser [18]. HDFS has 33 event types, 558,223 normal and 16,838 abnormal sequences (average length 29 ± 6); BGL has 394 event types, 37,823 normal and 31,429 abnormal sequences (average length 69 ± 747). The merged vocabulary has 427 events. Per-domain precision, recall, and F_1 (anomaly is the positive class), forgetting (F_1before $-$ F_1after), and model state size in bytes are reported. Seeds 0–4 are used where noted, and results are given as mean $\pm$ population standard deviation. Every headline effect has the same sign in *all* seeds and a very large paired effect size; the paired significance tests are collected in Section V-B (Table V). All experiments run on commodity CPU, and DeepLog evaluation is made tractable by deduplicating identical test sequences and weighting by count.

Before drawing any mixed-domain conclusion, the single-domain baselines are reproduced (Table II). The numbers match the reference picture. HDFS Length is weak on its own (0.538 versus the reference's $\sim$0.57), BGL Known Events is strong (0.992 versus $\sim$0.99), and every method is strong or weak on the datasets where it should be. The lightweight methods therefore pass the reproduction check.

TABLE II

Reproduction check: single-domain baselines match the reference picture.

A. *DeepLog operating point and sensitivity*

Domain	Method	F_1 (proposed)	Reference
HDFS	Length	0.5379	∼0.57 (Length alone)
HDFS	Known Events	0.7885	—
BGL	Length	0.0203	weak
BGL	Known Events	0.9915	∼0.99 (Events)

The deep-model results below depend on where DeepLog is operated. Its two hyperparameters, the top-g decision threshold and the number of federated rounds, are therefore fixed by direct measurement on single-domain HDFS before any mixed-domain experiment, and the sensitivity is reported so that the operating point is not a hidden degree of freedom. Table III sweeps the top-g rule. F_1 is highest at the reference's default g=9 (0.7039). Precision rises steadily with g, while recall is capped near 0.63 for every small g and then falls, so about a third of HDFS anomalies are predicted correctly by the next-event model and cannot be recovered by any threshold. The value g=9 is kept throughout. Table IV sweeps the number of rounds: F_1 reaches its plateau by round 5 and moves little afterwards (best 0.7205 at 100 rounds). The 12-round H2 and H3 runs therefore sit on the plateau, and the forgetting effect is not an under-training artifact. These two sweeps also bear on the discussion of absolute performance in Section XVI.

TABLE III

TOP-g SENSITIVITY OF SINGLE-DOMAIN HDFS DEEPLOG (PLATEAU MODEL, 12 ROUNDS). F_1 PEAKS AT THE REFERENCE DEFAULT g=9; RECALL IS CAPPED NEAR 0.63 FOR SMALL g, SO A THIRD OF ANOMALIES ARE UNRECOVERABLE BY ANY THRESHOLD.

top-g	Precision	Recall	F_1
1	0.033	0.632	0.063
5	0.558	0.632	0.593
7	0.785	0.632	0.700
9	0.806	0.625	0.704
12	0.880	0.538	0.668
15	0.962	0.461	0.623
20	0.953	0.086	0.158

TABLE IV

ROUND SENSITIVITY OF SINGLE-DOMAIN HDFS DEEPLOG (SCALAR ID, g=9). F_1 PLATEAUS BY ROUND 5; THE H2/H3 RUNS USE 12 ROUNDS.

B. *Statistical analysis*

Significance is quantified rather than relying on single runs. For each headline effect, measured over multiple seeds, a paired t-test between the two conditions, the paired effect size (Cohen's d_z), and whether the effect keeps the same sign in *every* seed are reported (Table V). All three headline effects (the H1 range collapse, the H3 forgetting, and the replay mitigation) are significant at $p < 10-4$, with very large effect

Rounds	Precision	Recall	F_1
1	0.047	0.596	0.087
5	0.806	0.625	0.704
12	0.806	0.625	0.704
25	0.782	0.630	0.698
50	0.796	0.580	0.671
100	0.852	0.624	0.721

sizes ($d_z > 20$) and the same sign in every seed, so none depends on a lucky run. Two caveats apply. First, the effects are so large (full F_1 swings of 0.5–0.7) that even small seed counts give tiny p-values, so the t-statistics and effect sizes are also reported, which show that the effect is large and not merely significant. Second, where variance is genuinely high, this is stated: domain-blind BGL Length and HDFS Known Events have wide seed spreads (Tables VI and VII), which is why no claim is built on them.

TABLE V

Statistical significance of the three headline effects (paired across seeds). All are significant with very large effect sizes and have the same sign in every seed.

Effect (paired comparison)	n	paired t	p	d_z
H1: domain-aware − blind (HDFS Length)	5	47.4	$1.2 \times 10{-}6$	> 20
H3: embedding − scalar forgetting	3	293	$1.2 \times 10{-}5$	> 20
Replay: no-buffer − 10% forgetting	3	196	$2.6 \times 10{-}5$	> 20

VI. H1: Range Aggregation Collapses under Domain Mixing

Table VII and Fig. 1 report per-domain F_1 over five seeds in the mixed federation. Domain-blind Length on HDFS collapses to $F_1 \approx 0.000$: BGL's large length spread (69 ± 747) widens the shared global range until almost no HDFS sequence looks anomalous. The domain-aware Length restores HDFS to 0.561 ± 0.024, statistically indistinguishable from its single-domain baseline (0.538, dashed line in Fig. 1). This is a direct counterexample to C1 under domain skew. The aggregation is invariant *within* a dataset, but *across* datasets it is dominated by the most variable domain.

TABLE VI

H1 is unchanged under a log-normal quantity skew (non-IID), so the domain-aware fix is partition-invariant. Per-domain F_1, mean $\pm$ std, $n = 5$ seeds.

TABLE VII

Method	Domain	IID F_1	non-IID F_1
Length, global (domain-blind)	HDFS	0.000 ± 0.000	0.000 ± 0.000
Length, domain-aware (proposed)	HDFS	0.561 ± 0.024	0.561 ± 0.024
Length, global (domain-blind)	BGL	0.033 ± 0.044	0.045 ± 0.051
Known Events (set union)	BGL	0.987 ± 0.003	0.991 ± 0.001

H1: PER-DOMAIN F_1 (MEAN ± STD, $n = 5$ SEEDS), IID MIXED FEDERATION.

Method	HDFS F_1	BGL F_1
Length, global (domain-blind)	0.0000 ± 0.0001	0.0332 ± 0.0435
Length, domain-aware (proposed)	0.5614 ± 0.0236	0.0332 ± 0.0435
Known Events (set union)	0.5636 ± 0.1170	0.9866 ± 0.0034
HDFS single-domain Length	0.5379	—

Three observations sharpen the result. First, the damage is *asymmetric*. BGL Length is *identical* under both aggregations in every seed, because BGL is the domain that *widens* the range rather than its victim, so the repair is one-sided. Second, the effect is *specific* to range-based aggregation: Known Events (set union) is unaffected by mixing, which makes it possible to point to a mechanism instead of asserting a general collapse. Third, the story is unchanged under a log-normal quantity skew, and domain-aware Length is *partition-invariant* by construction (Table VI). Under that skew the domain-aware HDFS number is identical to the IID value (0.561), and the domain-blind collapse is likewise unchanged, because a domain's aggregated range is the min–max over the union of its clients' data however that data is split. The non-IID setting only raises the variance of a few estimates (for example domain-blind BGL Length); it never changes the sign or the conclusion.

Why this generalizes: a containment condition. The collapse is a structural property of min–max union, not an accident of this domain pair, so the condition is

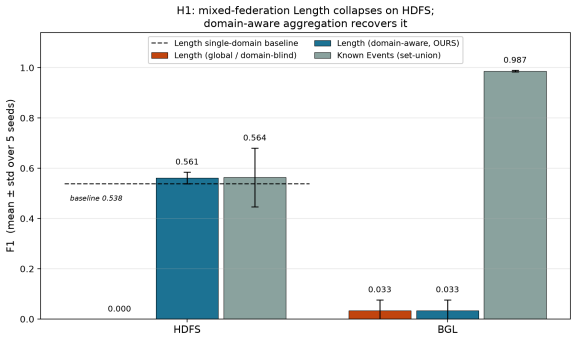


Fig. 1 H1: in the mixed federation, domain-blind Length collapses on HDFS ($F_1 \approx 0$), and domain-aware aggregation brings it back to the single-domain baseline (dashed). Known Events is unaffected, and BGL does not change under the fix, so the contamination and its repair are asymmetric.

stated formally. Domain-blind Length flags a test sequence of length ℓ if and only if $\ell \notin [\underline{L}, L]$, where $\underline{L} = \min_d \underline{L}_d$ and $L = \max_d L_d$ are taken over all co-training domains.

Proposition 1 (Range-aggregation collapse)**.** *Let domain a have normal-length range $[\underline{L}_a, L_a]$ and anomaly-length support A_a with $A_a \cap [\underline{L}_a, L_a] = \emptyset$ (so the single-domain range detector attains recall 1 on A_a). If some co-training domain $b \neq a$ satisfies $A_a \subseteq [\underline{L}_b, L_b]$, then domain-blind Length attains recall 0 (hence $F_1 = 0$) on a, whereas domain-aware Length attains the single-domain recall.*

Proof. Since $[\underline{L}, L]$ **[?]** $[\underline{L}_b, L_b]$ **[?]** A_a, every anomaly length $\ell \in A_a$ satisfies $\ell \in [\underline{L}, L]$, so the flag predicate $\ell \notin [\underline{L}, L]$ is false and no anomaly of a is flagged; with zero true positives, $F_1 = 0$. Domain-aware Length evaluates $\ell \notin [\underline{L}_a, L_a]$, which holds for every $\ell \in A_a$ by hypothesis, recovering the single-domain recall.

The hypothesis holds for *arbitrary* domains and any number of them: a single broad-length domain collapses range detection on *every* tighter co-resident. BGL (69 ± 747) contains HDFS's anomaly-length support (29 ± 6), which is exactly the measured $0.538 \rightarrow 0.000$ collapse, and Section XII-A confirms the proposition with a second co-resident (OpenStack) present. The domain-aware estimator removes the cross-domain coupling by construction, for any |D|.

Corollary 1 (Tag-free recovery)**.** *If the domains own disjoint event-id blocks, each sequence's domain is recoverable exactly from its event ids; hence domain-aware Length attains its single-domain F_1 with no external domain tag.*

The vocabulary-block router described above realizes Corollary 1; its routing accuracy is 1.000.

The fix needs no oracle tag. A real deployment may not label a sequence's domain, so the domain-aware fix is evaluated under the two tag-free routing strategies and both are compared with the oracle (Table VIII, five seeds). *Vocabulary-block routing* infers a sequence's domain from the event-id block its ids fall in (HDFS owns [0, 33), BGL [33, 427)), by majority, and flags a sequence whose ids fall outside every known block as anomalous. *Conservative routing* does no inference at all: it flags a sequence only if its length lies outside *every* domain's range. Vocabulary-block routing recovers the true domain of *every* one of the 569,479 HDFS and 68,874 BGL test sequences per seed (routing accuracy 1.000 ± 0.000, zero unknown-domain sequences),

and so reproduces the oracle-tag F_1 *exactly*, seed for seed (HDFS 0.5614 ± 0.0236, BGL 0.0332 ± 0.0435). Conservative routing, by contrast, collapses on HDFS (F_1 0.000 ± 0.000). HDFS anomalies are long enough to sit inside BGL's wide range, so "outside every range" never fires for them, which is the same containment that causes H1. The disjoint id blocks that *cause* the domain-blind collapse are thus the same structure that makes the domain recoverable at inference, and neither an external label nor a learned router is needed. This works because the two vocabularies are disjoint, so the event id *is* the domain label. The open case is *overlapping* vocabularies (a template shared across domains), where a block no longer identifies the domain uniquely; it is left to future work.

TABLE VIII

TAG-FREE ROUTING FOR DOMAIN-AWARE LENGTH (MEAN ± STD, $n = 5$ SEEDS). VOCABULARY-BLOCK INFERENCE EQUALS ORACLE ROUTING EXACTLY (ROUTING ACCURACY 1.000); CONSERVATIVE ROUTING NEEDS NO INFERENCE BUT COLLAPSES ON HDFS.

Domain	**Oracle F_1**	**Vocab-block F_1**	**Conservative F_1**
HDFS	0.5614 ± 0.0236	**0.5614 ± 0.0236**	0.0000 ± 0.0001
BGL	0.0332 ± 0.0435	0.0332 ± 0.0435	0.0332 ± 0.0435

VII. H2: SIMULTANEOUS MIXED FEDAVG DOES NOT DEGRADE DEEPLOG

Table IX compares single-domain and mixed DeepLog per domain, averaged over three seeds. On HDFS the drop is essentially zero (+0.0002 ± 0.0001), and on BGL it is consistently *negative*, meaning that mixed is better (−0.1058 ± 0.0023). Simultaneous mixed FedAvg thus does not degrade the deep model, which supports, rather than contradicts, the generality of deep federated detection reported by prior work. The reason lies in the input. With the reference's scalar-id input, HDFS ids occupy [0, 32] and BGL ids [33, 426] (disjoint numeric ranges), so the LSTM can separate the domains by input magnitude alone, and weight averaging never has to reconcile conflicting behaviours on the same inputs. The negative drop persists even under a learned embedding (HDFS drop −0.0025 ± 0.0018; Section IX), because every FedAvg round sees both domains and the averaged model maintains both. This robustness does not, however, extend to sequential arrival.

TABLE IX

H2: DEEPLOG PER-DOMAIN F_1, SINGLE-DOMAIN VS. MIXED (MEAN ± STD, $n = 3$ SEEDS).

Domain	**Single-domain**	**Mixed**	**Drop (single − mixed)**
HDFS	0.7039	0.7037	+0.0002 ± 0.0001
BGL	0.6459	0.7517	−0.1058 ± 0.0023

VIII. H3: SEQUENTIAL ARRIVAL, GROW-FOREVER VERSUS FORGET

Under sequential arrival (HDFS first, then BGL), the two families fail in complementary ways (Table X, Fig. 2). Known Events *never forgets*: the union is monotone, so HDFS F_1 stays at 0.7885. Its state, however, grows ×3.7 (1,688 → 6,300 bytes) as BGL's vocabulary is absorbed, and on an open-ended stream of domains that growth is unbounded, which contradicts C2 under evolution. DeepLog with the scalar-id input shows *no* forgetting (HDFS F_1 unchanged), for the same disjoint-range reason as in H2: continued training on BGL updates the [33, 426] input region and barely perturbs the [0, 32] region that HDFS occupies.

TABLE X

H3: SEQUENTIAL ARRIVAL (HDFS→BGL). FORGETTING = F_1 BEFORE − AFTER. THE KNOWN EVENTS AND DEEPLOG (SCALAR ID) ROWS ARE SINGLE-SEED (SEED 0) AND DETERMINISTIC; THE DEEPLOG (EMBEDDING) ROW REPORTS THE 3-SEED MEAN ± STD.

Method / input	**After HDFS**	**After BGL**	**Forgetting**
Known Events	0.7885	0.7885	0.0000
DeepLog (scalar id)	0.7037	0.7037	0.0000
DeepLog (embedding)	0.7037	0.039	+0.6647 ± 0.0032

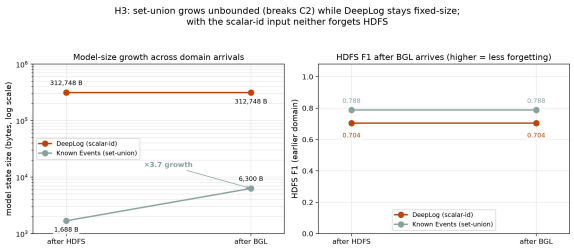


Fig. 2 H3: across domain arrivals, the set-union model grows (left; ×3.7) while DeepLog stays fixed-size. With the scalar-id input, neither forgets HDFS (right).

IX. MECHANISM: REPRESENTATION DECIDES FORGETTING

The H2 and H3 negatives above share one candidate cause: the scalar-id encoding keeps the two vocabularies in disjoint input ranges. This is tested directly by swapping DeepLog's scalar input for a learned 16-dimensional embedding, in which the vocabularies overlap in feature space, and re-running H2 and H3 over three seeds (Fig. 3). The result is clear. Under *sequential* arrival the embedding model *catastrophically forgets* HDFS: forgetting is 0.665 ± 0.003, with HDFS F_1 falling from 0.70 to 0.04, against 0.000 for the scalar input. Under *simultaneous* mixing the embedding model still does not degrade (HDFS drop −0.0025 ± 0.0018). The representation, not the aggregation method, therefore decides whether the deep model forgets.

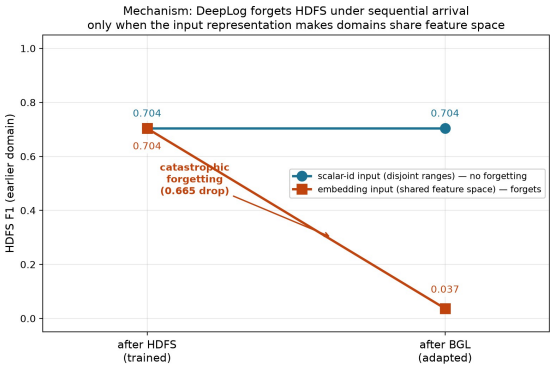


Fig. 3 Mechanism: under sequential arrival, DeepLog forgets the earlier domain only when the input representation makes the domains share a feature space (embedding, orange: 0.70 → 0.04). The scalar-id encoding (blue) keeps them in disjoint ranges, and nothing is forgotten.

The picture is a 2×2 over {simultaneous mixing, sequential arrival}×{disjoint-range input, shared-representation input} (Fig. 4). Only *(sequential, shared representation)* fails, and it fails hard. Going through the map cell by cell shows the mechanism:

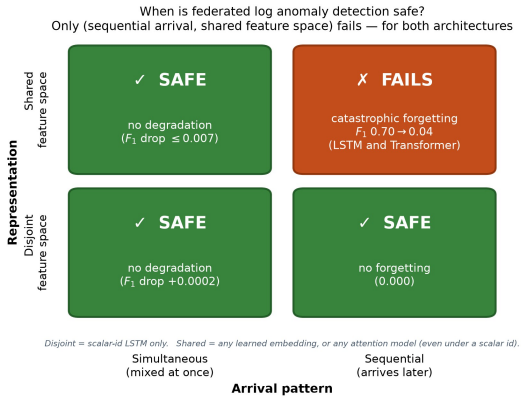


Fig. 4 The 2 × 2 safety map summarizing H2, H3, and the mechanism and architecture experiments. Across {arrival pattern}×{feature space}, only (sequential arrival, shared feature space) fails, for both an LSTM and a Transformer; the other three regimes are safe. A disjoint feature space arises only for a scalar-id LSTM, since any learned embedding, and any attention model (even under a scalar id), shares one. The failing cell applies to the forward order HDFS→BGL and is directional (Table XI).

- *Simultaneous* × *disjoint feature space* (scalar-id LSTM): **safe**. Both domains appear in every FedAvg round and their ids occupy disjoint numeric ranges, so the model separates them by input magnitude and never has to reconcile conflicting behaviour on the same inputs (H2, drop +0.0002).

- *Simultaneous* × *shared feature space* (embedding or attention): **safe**. The vocabularies now overlap in feature space, but both domains are present in every round, so the averaged model maintains both (H2 embedding, HDFS drop −0.0025).
- *Sequential* × *disjoint feature space* (scalar-id LSTM): **safe**. Training on the new domain updates only the input region that domain occupies and barely perturbs the earlier one, so nothing is forgotten (H3 scalar, forgetting 0.000).
- *Sequential* × *shared feature space* (embedding or attention): **catastrophic**. Training on the new domain dominates the shared representation and overwrites the region the earlier domain used, which erases its detection (H3 embedding, forgetting 0.665; and both Transformer inputs, Section XII-B).

Simultaneous mixing is therefore safe across the board; the danger lies in onboarding a domain sequentially into a shared-representation model. This distinction explains why the reference's scalar-id DeepLog appears robust. It also suggests that any semantics-based encoding (embeddings, template vectors, or large language model (LLM) features) will need an explicit continual-learning defense [17], [19], since such encodings necessarily share feature space across domains.

Forgetting is directional. A natural follow-up is whether the catastrophic cell is symmetric in arrival order. The order is reversed (BGL first, then HDFS) and the amount of the earlier domain (now BGL) that is retained is measured, over three seeds (Table XI). It is *not* symmetric: in the reverse order the embedding model does *not* forget BGL. BGL F_1 actually rises from 0.649 to 0.752 ("forgetting" -0.103 ± 0.006), and the scalar input is likewise safe (-0.125 ± 0.017). So the failure is not merely *(sequential, shared representation)*; it is specifically *onboarding the broad-vocabulary domain (BGL, 394 events) after the narrow one (HDFS, 33 events)*. The larger new domain's training dominates the shared representation and overwrites the narrow region the earlier domain occupied. In the reverse case a small vocabulary arrives into a model that already spans a broad one and perturbs it little. This turns the failing cell into a concrete, testable direction, and it suggests that onboarding order is itself a deployment lever: where possible, introduce broad-vocabulary domains *earlier*.

TABLE XI

ARRIVAL ORDER IS NOT SYMMETRIC. IN THE REVERSE ORDER (BGL→HDFS) THE EARLIER DOMAIN IS NOT FORGOTTEN UNDER EITHER INPUT, IN CONTRAST TO THE FORWARD EMBEDDING CASE (HDFS→BGL), WHICH FORGETS +0.665. MEAN ± STD, $n = 3$ SEEDS.

Order / input	**Earlier before**	**Earlier after**	**Forgetting**
HDFS→BGL, embedding	0.704	0.039 ± 0.003	+0.665 ± 0.003
BGL→HDFS, embedding	0.649	0.752 ± 0.000	−0.103 ± 0.006
BGL→HDFS, scalar id	0.627	0.752 ± 0.000	−0.125 ± 0.017

X. MITIGATION: REPLAY RESTORES THE FORGOTTEN DOMAIN

The mechanism identifies *(sequential arrival, shared representation)* as the single failing cell, and a diagnosis is only useful if the failure can be fixed. The simplest continual-learning defense, experience replay [19], is tested. When the new domain (BGL) arrives, a small buffer of the earlier domain's (HDFS) training sequences is kept and training is performed on BGL and the buffer jointly, as one extra federated client, instead of on BGL alone. The buffer is a p-fraction sample of the pooled HDFS training set (5,582 sequences, itself the reference's 1% regime). Everything else (embedding input, rounds, seeds) is unchanged from Section IX. Because replay stores raw sequences and not parameters, the model stays fixed-size.

Table XII and Fig. 5 give the result over three seeds. Without replay the embedding model forgets HDFS (F_1 0.70 $\rightarrow$ 0.04, forgetting 0.665 $\pm$ 0.003, reproducing Section IX). A buffer of just 10% ($\approx$ 558 sequences) *eliminates* the forgetting: HDFS returns to 0.700 $\pm$ 0.004, its pre-arrival level, while BGL is still learned (0.722 $\pm$ 0.052). A 25% buffer is cleaner still (forgetting 0.001 $\pm$ 0.001, BGL 0.758 $\pm$ 0.009). A 5% buffer is too small to be reliable: it recovers HDFS in only one of three seeds, giving forgetting 0.440 $\pm$ 0.308, which sets a floor on the buffer size. The failure is real but cheap to remedy, needing a few hundred stored sequences, no architecture change, and no loss on the new domain. If sequential onboarding into a shared-representation model is unavoidable, budget a small replay buffer.

TABLE XII

REPLAY MITIGATION FOR EMBEDDING-DEEPLOG UNDER SEQUENTIAL ARRIVAL (HDFS$\rightarrow$BGL), MEAN $\pm$ STD OVER $n = 3$ SEEDS. BUFFER $= p\times$ THE 5,582-SEQUENCE HDFS TRAINING POOL. A $\geq$ 10% BUFFER RESTORES HDFS TO ITS PRE-ARRIVAL F_1 (0.704) WHILE BGL IS STILL LEARNED.

Buffer	**HDFS F_1 after**	**BGL F_1 after**	**Forgetting**
none (reproduce)	0.039 $\pm$ 0.003	0.642 $\pm$ 0.028	+0.665 $\pm$ 0.003
5% (279 seq)	0.264 $\pm$ 0.308	0.651 $\pm$ 0.002	+0.440 $\pm$ 0.308
10% (558 seq)	**0.700 $\pm$ 0.004**	0.722 $\pm$ 0.052	**+0.004 $\pm$ 0.004**
25% (1396 seq)	0.702 $\pm$ 0.001	0.758 $\pm$ 0.009	+0.001 $\pm$ 0.001
50% (2791 seq)	0.694 $\pm$ 0.014	0.752 $\pm$ 0.000	+0.009 $\pm$ 0.014

A. *A memory-free alternative: EWC*

Replay stores raw sequences of the earlier domain, which some deployments cannot do (data-retention limits, or a client that has left the federation). The standard *parameter-based*, memory-free defense, EWC [17], is therefore also evaluated under the identical protocol. From the stage-1 HDFS model, the empirical Fisher information F_i of each weight on HDFS windows is estimated, and federated training on BGL then continues with the penalty $(\lambda/2)\ \Sigma_i\ F_i(\theta_i - \theta^*{}_i)2$, which anchors the HDFS-important weights θ^*. Table XIII sweeps λ over three seeds. EWC *does* remove the forgetting: at λ=108 it restores HDFS to 0.690 $\pm$ 0.005 (forgetting 0.014 $\pm$ 0.005) while BGL is still learned (0.752 $\pm$ 0.052), matching replay's outcome without storing any data. The catch is tuning. The penalty has no effect until λ=106 and only succeeds at λ=108, so EWC must be tuned across six orders of magnitude, whereas replay works out of the

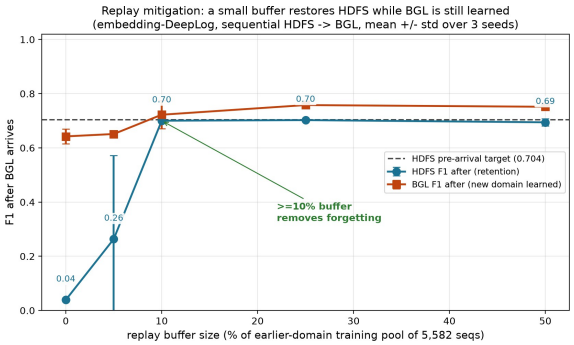


Fig. 5 Replay mitigation: HDFS retention and BGL learning versus buffer size (mean ± std, three seeds). Without a buffer HDFS collapses to 0.04. A buffer of $\geq$ 10% restores it to its pre-arrival level (dashed) while BGL is still learned. The 5% point is unreliable (large variance), which sets the minimum useful buffer.

box at a 10% buffer. The takeaway is a trade-off, not a single winner: prefer replay when a small earlier-domain buffer is permissible (no tuning), and EWC when it is not (no storage, but a sensitive hyperparameter to calibrate).

TABLE XIII

MEMORY-FREE EWC FOR EMBEDDING-DEEPLOG UNDER SEQUENTIAL ARRIVAL (HDFS→BGL), MEAN ± STD OVER $n = 3$ SEEDS. EWC REMOVES THE FORGETTING ONLY AT λ=108; SMALLER λ BARELY HELPS.

EWC λ	**HDFS F_1 after**	**BGL F_1 after**	**Forgetting**
0 (no defense)	0.039 ± 0.003	0.642 ± 0.028	+0.665 ± 0.003
102	0.040 ± 0.004	0.649 ± 0.019	+0.664 ± 0.004
104	0.044 ± 0.010	0.651 ± 0.015	+0.659 ± 0.010
106	0.212 ± 0.174	0.663 ± 0.047	+0.491 ± 0.174
108	**0.690 ± 0.005**	0.752 ± 0.052	**+0.014 ± 0.005**

XI. DOES THE FIX LIFT THE FULL ENSEMBLE?

The reference's strongest HDFS method is the union ensemble Events ∨ Length ∨ ECVC. It is reproduced first: the single-domain HDFS ensemble reaches $F_1 = 0.984$ (the reference reports ∼0.95). In the mixed federation (Table XIV), swapping domain-blind for domain-aware Length leaves the ensemble F_1 *unchanged*. The reason is structural. The combination is a logical OR, so a collapsed Length cannot lower recall, and on HDFS ECVC alone already reaches $F_1 = 0.968$ (recall 0.96), which pushes the union's recall to ≈ 1.0. A recovered Length only adds detections the ensemble already makes. The practical impact of H1 is therefore scoped accordingly. Domain-aware aggregation matters for *lightweight-only* deployments (Length and Events without a costly nearest-neighbour store, which is the efficiency regime that [3] motivates), where HDFS Length moves from 0.000 to 0.561. It does not change the full ensemble, which is carried by ECVC. No claim is made that domain mixing breaks the reference's best method.

TABLE XIV

ENSEMBLE EVENTS ∨ LENGTH ∨ ECVC: HDFS F_1 IS UNCHANGED BY THE LENGTH FIX BECAUSE ECVC DOMINATES THE UNION.

Configuration	**HDFS ensemble F_1**
Single-domain (reference target)	0.9836
Mixed, Length global (collapsed 0.0002)	0.9837
Mixed, Length domain-aware (recovered 0.5379)	0.9836

A. *Comparison to an established method*

Replay is deliberately the simplest possible defense, so a fair question is whether a purpose-built continual-learning method does better. The comparison is against A-GEM [20], a standard episodic-memory method, at the *same* 10% memory budget. Instead of training on the buffer jointly, A-GEM projects each new-domain gradient so that it cannot increase the loss on a memory batch of the earlier domain. Table XV puts the defenses side by side. All three remove the forgetting (HDFS retention $\approx$ 0.69–0.70 versus 0.04 undefended). At the matched budget the plain replay is at least as good as A-GEM on retention and forgetting (+0.004 versus +0.006) and, because A-GEM's projection constrains the new-domain update, learns BGL *better* (0.722 versus 0.657). The memory-free EWC learns BGL best but needs the six-order-of-magnitude penalty search of Section X-A and forgets slightly more. The simple fix is thus Pareto-competitive with the established toolkit: replay dominates the more complex A-GEM at equal memory, and trades cleanly against EWC (no tuning and less forgetting, at the cost of a small buffer). No exotic machinery is needed to make sequential onboarding safe.

TABLE XV

MITIGATION COMPARISON AT A MATCHED 10% MEMORY BUDGET (EMBEDDING-DEEPLOG, SEQUENTIAL HDFS→BGL, MEAN ± STD, $n = 3$). ALL THREE DEFENSES REMOVE THE FORGETTING; THE SIMPLE REPLAY MATCHES THE ESTABLISHED A-GEM ON RETENTION AND FORGETTING WHILE LEARNING THE NEW DOMAIN BETTER.

Method	**Mem.**	**HDFS after**	**BGL after**	**Forgetting**
No defense	—	0.039 ± 0.003	0.642 ± 0.028	+0.665 ± 0.003
EWC (λ=108)	none	0.690 ± 0.005	**0.752 ± 0.052**	+0.014 ± 0.005
A-GEM [20]	10%	0.698 ± 0.002	0.657 ± 0.009	+0.006 ± 0.002
Replay (proposed)	10%	**0.700 ± 0.004**	0.722 ± 0.052	**+0.004 ± 0.004**

XII. Generality: More Domains and a Modern Architecture

A study of two datasets and one detector raises the question of whether the map describes federated log analytics in general or only this pair and model. Both axes are tested directly, by adding a third and a fourth domain and a categorically different architecture.

A. *A third and fourth log domain*

OpenStack [4] is added as a third domain. It is parsed from the Loghub [7] raw logs into the same event-id-sequence format (Drain templates, grouped per VM instance) and occupies a third disjoint vocabulary block [427, 443), for a merged 443-event space. OpenStack is deliberately different in profile. Its vocabulary is a narrow 16 events, its normal length is very tight (24.9 ± 0.8), and its labelled anomalies are *order*-based (identical event set and length to normal sequences), so only a sequence model can detect them. This gives a third anomaly type, distinct from HDFS (length and event) and BGL (event). H1 and tag-free routing are rerun on the mixed federation {HDFS, BGL, OpenStack} over five seeds (Table XVI, Fig. 6). Three results stand out. First, the H1 collapse and recovery on HDFS is *numerically identical* to the two-domain case (Length $0.000 \rightarrow 0.561$) even with OpenStack also co-resident, which confirms Proposition 1 beyond the pair. Second, vocabulary-block routing recovers the true domain of *every* test sequence across all three disjoint blocks (accuracy 1.000; 569,479 + 68,874 + 1,857 sequences), so Corollary 1 holds at three domains. Third, OpenStack's order-based anomalies evade *both* lightweight detectors (Length and Known Events = 0). This reinforces that no single lightweight statistic dominates and that such domains require the deep model, while the tight normal set makes the length-range result solid. OpenStack ships only four labelled anomalies (a documented benchmark limitation), so it is used for the range and routing generality result and its positive count is reported openly instead of being presented as a headline detection score.

TABLE XVI

H1 and routing generalize to a three-domain federation {HDFS, BGL, OpenStack} (mean ± std, $n = 5$ seeds). HDFS collapse and recovery is identical to the two-domain case; routing is exact across three disjoint blocks.

Domain	**Length blind**	**Length d.-aware**	**Known Events**	**Route acc.**
HDFS	0.000 ± 0.000	$\mathbf{0.561 \pm 0.024}$	0.564 ± 0.117	1.000
BGL	0.033 ± 0.044	0.033 ± 0.044	0.987 ± 0.003	1.000
OpenStack	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	1.000

A fourth domain, and a relative-containment test. Hadoop [7] is added as a fourth domain (55 applications, 258 event types, parsed per application). It was chosen because its normal-length range is *broad*: its maximum (3,666) exceeds BGL's (900). Proposition 1 then makes a non-trivial prediction. In the four-domain federation {HDFS, BGL, OpenStack, Hadoop}, Hadoop becomes the new widener and should swallow *BGL*, which was itself the widener (and hence safe) in the two- and three-domain cases. Fig. 7 confirms this. HDFS collapse and recovery is unchanged ($0.000 \rightarrow$

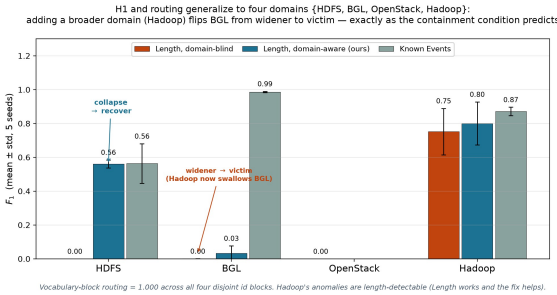


Fig. 6 H1 and tag-free routing generalize to a three-domain federation. HDFS collapse and recovery (domain-blind 0.00 → domain-aware 0.56) is numerically identical to the two-domain case even with OpenStack co-resident, and vocabulary-block routing is exact across all three disjoint id blocks. OpenStack's order-based anomalies are invisible to both lightweight detectors, which marks it as a distinct third profile.

0.561), routing stays exact across all four disjoint blocks (1.000), and BGL's domain-blind Length drops from 0.033 to 0.001, so BGL *flips from widener to victim*, while domain-aware aggregation recovers it. Hadoop's own anomalies are length-detectable (Length 0.75, domain-aware 0.80), a fourth, different profile. The containment relation is therefore *relative*: a domain is safe only until a broader one joins, and the proposition predicts which one is affected for any membership. (Because Hadoop ships only 11 normal applications, it uses a larger training fraction than the 1% regime; this is reported openly.)

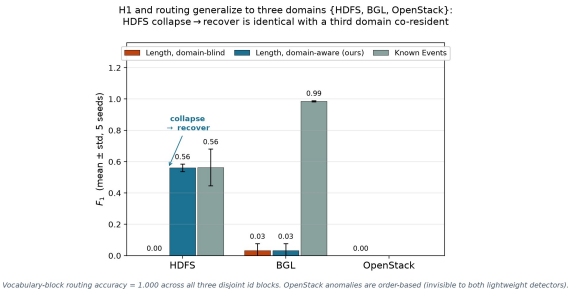


Fig. 7 Four domains and a relative-containment test. Adding Hadoop, whose normal length exceeds BGL's, makes it the new widener: BGL's domain-blind Length drops 0.033 → 0.001, so BGL flips from widener to victim, as Proposition 1 predicts. HDFS collapse and recovery is unchanged, and routing stays exact across four disjoint blocks. Hadoop's anomalies are length-detectable.

B. *A modern architecture: attention forgets too*

To test whether the forgetting is an artifact of DeepLog's recurrence, the LSTM is replaced with a small self-attention Transformer (two encoder layers, 64-d, four heads, learned positional encoding) that consumes the identical windows and top-g rule. H2 (simultaneous) and H3 (sequential) are rerun over three seeds under both input encodings (Table XVII, Fig. 8). The map holds. Under *simultaneous* mixing the Transformer does not degrade (HDFS drop −0.022 scalar, −0.007 embedding; BGL

improves), whereas under *sequential* arrival it *catastrophically forgets* HDFS (forgetting 0.636 scalar, 0.361 embedding, with HDFS F_1 falling from ∼0.68). Two points sharpen the mechanism. First, the failure is *not* specific to DeepLog, because a categorically different architecture reproduces it. Second, the Transformer forgets even under the *scalar* input that left the LSTM immune (Table X). Its learned input projection maps the scalar id into a shared d-dimensional space, so the two vocabularies overlap in feature space whatever the raw encoding. This indicates that a *shared feature space*, and not the embedding layer itself, is the operative cause, and that the LSTM's scalar-input immunity is an architecture-specific accident. Modern encoders, which project their inputs by design, are *more* exposed, not less, which is the model-class-wide risk discussed in Section XIII.

TABLE XVII

THE FORGETTING IS ARCHITECTURE-GENERAL. UNDER THE IDENTICAL PROTOCOL, A SELF-ATTENTION TRANSFORMER IS SAFE UNDER SIMULTANEOUS MIXING BUT CATASTROPHICALLY FORGETS UNDER SEQUENTIAL ARRIVAL. IT FORGETS EVEN UNDER THE SCALAR INPUT THAT LEFT THE LSTM (TABLE X) IMMUNE, BECAUSE ITS INPUT PROJECTION SHARES FEATURE SPACE ACROSS DOMAINS. MEAN ± STD, n = 3 SEEDS.

Transformer input	**H2 HDFS drop (simult.)**	**H3 forgetting (sequential)**
scalar id	−0.022 ± 0.020	**+0.636 ± 0.003**
embedding	−0.007 ± 0.004	**+0.361 ± 0.140**

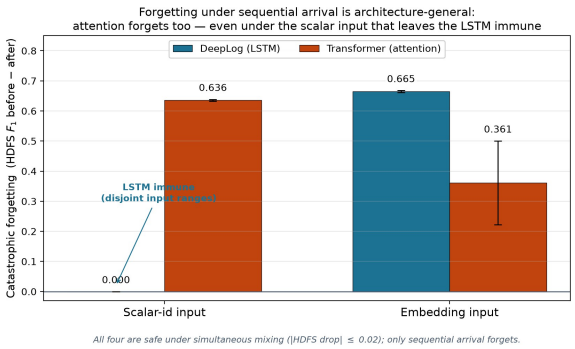


Fig. 8 Forgetting under sequential arrival is architecture-general. Of the four architecture×input combinations, only DeepLog with the scalar input is immune (disjoint input ranges). The Transformer forgets under both inputs, because its learned input projection places even the scalar id into a shared feature space. All four are safe under simultaneous mixing.

XIII. DEPLOYMENT IMPLICATIONS FOR PRODUCTION FEDERATIONS

Federated log analytics is adopted where centralizing logs is legally or operationally impossible: multi-tenant public clouds, banks and payment networks, healthcare

providers, telecom and IoT fleets, and critical-infrastructure operators. These organizations do not run one homogeneous log source. They run *many* categorically different ones (compute, storage, network, application, security appliances), added incrementally as services are onboarded. That is the heterogeneous, evolving regime this study targets, and the failures identified here have three properties that make them matter in production.

The failures are silent and security-relevant. Neither the range-aggregation collapse nor the catastrophic forgetting raises an error, shows up as a visibly bad loss curve, or crashes a service. The detector simply stops flagging anomalies for one domain while continuing to report healthy operation for the others. Fig. 9 shows the size of the loss: domain-blind mixing drops HDFS Length recall from 0.369 to 0.0001, so of the 6,211 HDFS anomalies the detector caught alone, all but about two silently stop being flagged (6,209 of 16,838 newly missed, 99.97% of the detector's coverage), while sequential onboarding erases 94.5% of the deep model's HDFS detection (F_1 0.70 $\rightarrow$ 0.04). In a security setting each missed anomaly is an intrusion that becomes invisible, and in a reliability setting an outage precursor is lost. A silent blind spot in a monitoring system is more dangerous than an outage of it, because operators keep trusting a system that has quietly stopped working.

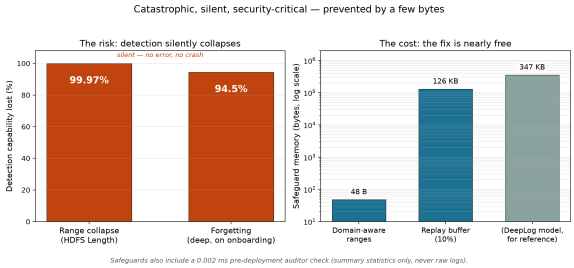


Fig. 9 The practical case in one view. Left: when a failure fires, detection silently collapses (99.97% of the range detector's coverage, 94.5% of the deep model's) with no error or crash. Right: the safeguards that prevent it cost anywhere from a few bytes (per-domain ranges) to a buffer smaller than the model it protects, plus a sub-millisecond pre-deployment check.

Scaling coverage can destroy coverage. The most worrying case is sequential onboarding. Adding a monitored source is a routine and frequent operations action, yet it can catastrophically erase detection for sources already in the federation (H3, Section IX); extending monitoring to a new service can silently blind monitoring of existing ones. The directionality result (Table XI) gives a concrete operational lever: onboarding a broad, high-cardinality source *after* a narrow one is the dangerous order, so where the schedule is controllable, broad domains should be introduced first.

The risk covers a whole class of models. The forgetting is shown on both an LSTM (DeepLog) and a self-attention Transformer (Section XII-B), and it is governed by whether the input representation shares a feature space across domains. Any semantics-based encoder (learned embeddings, template-vector models, or the LLM-derived log features the field is moving toward) shares feature space by construction

and is therefore exposed. As log analytics adopts large-model representations, this failure mode becomes *more* likely, not less.

The safeguards cost very little. The remedies are cheap enough to adopt unconditionally (Fig. 9, right). Per-domain range parameters add $O(|\mathrm{D}|)$ state (48 bytes for three domains). Tag-free vocabulary-block routing needs no labels (Corollary 1). A replay buffer of a few hundred sequences ($\approx$ 126 KB, *smaller than the* 347 KB *model it protects*) removes the forgetting without changing the architecture (Section X), and the pre-deployment audit itself runs in 0.002 ms. The comparison is the point: a silent, security-relevant failure is averted for a few bytes and a sub-millisecond check, so there is no efficiency-versus-safety trade-off to agonize over. The safety auditor (Section XIV-A) turns a latent *runtime* risk into a *pre-deployment* configuration check that uses only per-client summary statistics (event-id blocks and length ranges, never raw logs). A federation operator can therefore gate a rollout in continuous integration without weakening the privacy model that motivated federation in the first place. The net effect is that a realistic heterogeneous deployment can keep the efficiency and privacy benefits of federated log analytics without the silent failures that would otherwise accompany them.

XIV. Discussion

The results refine the two published claims rather than overturn them. C1 (invariance) *fails* for range-based aggregation under domain skew, and a per-domain fix restores it; it survives for set-union. C2 (bounded growth) *fails* for set-union under sequential arrival (unbounded vocabulary growth), while the deep model stays fixed-size. For the deep model, what matters in practice is not mixing but *arrival pattern combined with representation*: simultaneous FedAvg is safe, but sequential onboarding into a shared-representation model forgets catastrophically. Three pieces of practical guidance follow: (i) use per-domain parameters for any range or threshold statistic in a heterogeneous federation; (ii) prefer simultaneous over sequential onboarding when possible; and (iii) if sequential onboarding is unavoidable with a semantics-based model, budget a small replay buffer, which removes the forgetting at the cost of a few hundred stored sequences and no architecture change (Section X).

A. *A deployable safety auditor*

These three pieces of guidance are decision rules over quantities a federation already knows before training: each domain's event-id block (hence whether vocabularies are disjoint) and its normal-length range (hence the containment condition of Section VI), plus the intended arrival pattern and input representation. The map is therefore compiled into a small *pre-deployment auditor*. It takes only these per-client summary statistics (never raw logs, which preserves the federated privacy model) and returns, for a planned federation, which failures it will hit (range-aggregation collapse, unbounded set-union growth, or directional catastrophic forgetting) together with the prescribed fix (per-domain ranges, a bounded vocabulary, or a replay buffer or EWC). The auditor is deterministic, dependency-free, and released as a command-line tool alongside reference implementations of the fixes, so a practitioner can run it in seconds. To the best of the authors' knowledge, it is the first such safety check for heterogeneous federated log analytics.

B. *The auditor's predictions are validated*

An auditor is only useful if it is *right*. It is therefore run *blind* on every configuration for which a measured outcome exists (two architectures, two input encodings, three arrival patterns for the deep model, and the lightweight range-aggregation cases across the two- and three-domain federations), and its verdict is scored against what actually happened (Table XVIII). The auditor is correct in 14 of 15 configurations (93%), including *all ten* deep-model cells: it predicts exactly which (architecture, input, arrival) combinations forget, including that an attention model shares feature space even under a scalar id. Its single error is a *conservative false positive*. On OpenStack it flags the range-aggregation coupling, which is structurally real because OpenStack's tight length range is contained in the global one, even though OpenStack's anomalies happen to be order-based, so no length detector existed there to break. Erring toward flagging a real coupling is the safe direction for a pre-deployment check, and no genuine failure is missed. The auditor is thus not just a restatement of the findings but a *tested predictive instrument*.

TABLE XVIII

THE AUDITOR, RUN BLIND, PREDICTS THE MEASURED OUTCOME IN 14/15 CONFIGURATIONS. DEEP-MODEL CELLS SPAN 2 ARCHITECTURES × 2 INPUTS × ARRIVAL PATTERNS; LIGHTWEIGHT CELLS ARE THE DOMAIN-BLIND LENGTH COLLAPSE ACROSS 2- AND 3-DOMAIN FEDERATIONS.

Configuration group	**Configs**	**Auditor correct**
Deep-model forgetting (LSTM & Transformer)	10	**10**
Lightweight range collapse (2- & 3-domain)	5	4
Total	15	**14 (93%)**

XV. REPRODUCIBILITY

Every number in this paper can be regenerated on a laptop. **Code and environment.** The implementation is pure Python (3.12) and CPU-only, with no GPU dependency; the deep models use a small PyTorch stack, and the lightweight detectors use only NumPy. The detector definitions and the FedAvg protocol of the reference pipeline [3] are adopted, and its per-dataset dictionary is extended into the merged cross-domain vocabulary of Section V. A single driver script re-runs the full set of experiments. The safety auditor of Section XIV-A ships as a dependency-free command-line tool, together with reference implementations of the domain-aware aggregation, the vocabulary-block router, and the replay and EWC defenses. **Data.** All datasets are public from Loghub [7]: HDFS and BGL for the main study, OpenStack and Hadoop for the generality tests. Raw logs are parsed into event-id sequences with Drain [18], using the parameters stated in Section V. HDFS is grouped by block id, OpenStack by VM instance, and Hadoop by application.

Determinism and seeds. Every run is seeded explicitly. Lightweight experiments use seeds 0–4 and deep-model experiments seeds 0–2 (the deep runs are costlier), and for a given seed the pipeline is deterministic, so per-seed values reproduce exactly. The mean ± population standard deviation and the paired statistics of Section V-B are reported.

Traceability. Every table and figure in this paper is generated directly from the output of the experiment scripts; no reported value was entered by hand or estimated.

XVI. Threats to Validity

DeepLog absolute level. The single-domain HDFS DeepLog operates at $F_1 \approx 0.70$ (best 0.7205 over a 1–100 round sweep, Table IV), not at the reference's headline $\sim$0.91–0.93. Neither extra training nor tuning the top-g threshold reaches 0.93: F_1 plateaus by round 5, and recall is capped at $\sim$0.63 for every g (Table III), meaning roughly a third of HDFS anomalies are predicted perfectly by the next-event model and are unrecoverable by any threshold. The reference's *federated* configuration itself trains for only one round of one epoch on 1% of normals, so the value obtained is consistent with its federated regime; the $\sim$0.93 figure is most plausibly the reference's centralized or best-tuned result. To test the capacity ceiling directly, a standalone *centralized* DeepLog was run on the *full* HDFS normal set (all 558,223 sequences, 6.67M training windows, no federation). The run was budgeted for up to 100 passes, but it had fully converged by round $\sim$30 and was stopped at 55. It likewise does not approach 0.93: training loss plateaus at ≈ 0.204 by round 20 (and drifts only to 0.203 by round 55), and the transductive F_1 peaks at 0.43 (round 2) before settling near 0.25 (mean 0.257 ± 0.016 over rounds 10–55), with precision saturating at ≈ 0.98 while recall never exceeds 0.29. This indicates that the ceiling is a property of the scalar-input next-event model and the top-g rule, and not an artifact of federation or of 1% subsampling. No claim is therefore made to reproduce 0.93, and all H2 and H3 DeepLog numbers are for the federated, 1%-data regime. The H3 forgetting finding is a *relative* effect ($0.70 \rightarrow 0.04$ under embedding versus $0.70 \rightarrow 0.70$ under scalar) and does not depend on the absolute level; H2 and H3 were run at 12 rounds, already on the plateau.

Other threats. Four domains are used (HDFS, BGL, OpenStack, Hadoop). The range and routing results hold with third and fourth domains co-resident (Section XII-A), and the forgetting holds across two architectures (Section XII-B), so the findings are not artifacts of a single pair or model. The two added domains are small (OpenStack ships only four labelled, order-based anomalies; Hadoop only 55 applications, so it uses a larger-than-1% training fraction). They are therefore used for the range and routing generality and the relative-containment test, not as headline detection scores, and their sizes are reported openly. Scaling to many domains and to alert-labelled sources (Thunderbird, Spirit), which require a different windowing pipeline, remains future work. Known Events shows high seed variance on HDFS (0.56 ± 0.12), which is reported openly. The DeepLog evaluation treats sequences shorter than the window as normal; because the single-versus-mixed comparison uses an identical harness, the relative effect is unaffected. ECVC uses a transductive (test-set) threshold, following the reference convention. Finally, like [3], comparisons are made only within a controlled codebase with averaged runs, and no cross-study metric comparisons are made.

XVII. Conclusion and Future Work

The proposed system tested whether the two reported properties of federated log anomaly detection survive a heterogeneous and evolving federation. For range-based aggregation they do not: it collapses under domain mixing, and a minimal domain-aware aggregation restores single-domain performance. The deep model is robust to simultaneous mixed FedAvg but forgets catastrophically when a domain is onboarded sequentially into a shared-representation model, a danger that the reference's scalar-id encoding happens to mask. Set-union detectors never forget, but they grow without bound. What the proposed works contribute is a reproducible account of how failure depends jointly on the detection method, the arrival pattern, the input representation, and the onboarding *direction*. The proposed works made that account precise as a containment condition (Proposition 1), showed that it holds across three domains and two architectures, and turned it into working fixes (domain-aware aggregation with tag-free routing; replay and EWC for the forgetting) and a pre-deployment safety auditor. Because these failures are silent and matter for security in real federations, the practical upshot is a way to keep the privacy and efficiency of federated log analytics in a heterogeneous, evolving deployment without the blind spots that would otherwise come with it. Future work includes (i) longer arrival streams of many domains, where replay-buffer sizing, EWC λ scheduling, and cross-domain interference compound; (ii) alert-labelled sources (Thunderbird, Spirit) that need a different windowing pipeline and richer anomaly sets than OpenStack provides; and (iii) automatic selection of the EWC penalty, whose six-order-of-magnitude sensitivity is its main practical drawback relative to replay. **Data availability** All datasets are publicly available from Loghub[19] (https://github.com/logpai/loghub).

References

[1] M. Landauer, S. Onder, F. Skopik, and M. Wurzenberger, "Deep learning for anomaly detection in log data: A survey," *Mach. Learn. Appl.*, vol. 12, p. 100470, 2023.

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist. (AISTATS)*, 2017, pp. 1273–1282.

[3] A. Pustozerova, A. García Gómez, M. Landauer, M. Wurzenberger, F. Skopik, E. Weippl, R. Mayer, and A. Ekelhart, "Lightweight techniques for federated anomaly detection in log data," *IEEE Trans. Rel.*, vol. 75, 2026, doi: 10.1109/TR.2026.3677674.

[4] M. Du, F. Li, G. Zheng, and V. Srikumar, "DeepLog: Anomaly detection and diagnosis from system logs through deep learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2017, pp. 1285–1298.

[5] W. Meng *et al.*, "LogAnomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs," in *Proc. 28th Int. Joint Conf. Artif. Intell. (IJCAI)*, 2019, pp. 4739–4745.

[6] M. Landauer, F. Skopik, and M. Wurzenberger, "A critical review of common log data sets used for evaluation of sequence-based anomaly detection techniques," *Proc. ACM Softw. Eng.*, vol. 1, no. FSE, pp. 1354–1375, 2024, doi: 10.1145/3660768.

[7] J. Zhu, S. He, P. He, J. Liu, and M. R. Lyu, "Loghub: A large collection of system log datasets for AI-driven log analytics," in *Proc. IEEE 34th Int. Symp. Softw. Rel. Eng. (ISSRE)*, 2023. (arXiv:2008.06448.)

[8] J. Tian, M. Li, Z. Wang, L. Chen, J. Qin, and R. Zhang, "OMLog: Online log anomaly detection for evolving system with meta-learning," *arXiv preprint arXiv:2410.16612*, 2024.

[9] H. Xiang, G. Wang, Y. Xiao, F. Di, R. Gao, Y. Zhang, and X. Han, "Reliable and secure anomaly detection in heterogeneous federated learning: A comprehensive review," *Big Data Min. Anal.*, vol. 9, no. 3, pp. 821–840, 2026, doi: 10.26599/BDMA.2025.9020067.

[10] P. Himler, M. Landauer, F. Skopik, and M. Wurzenberger, "Anomaly detection in log-event sequences: A federated deep learning approach and open challenges," *Mach. Learn. Appl.*, vol. 16, p. 100554, 2024.

[11] Y. Liao, J. Keung, Z. Mao, J. Zhang, and J. Li, "FedLAD: A modular and adaptive testbed for federated log anomaly detection," *arXiv preprint arXiv:2512.08277*, 2025.

[12] Z. Gu, W. Yue, and C. Liu, "Explainable anomaly detection for system logs in distributed environments," *Comput., Mater. Continua*, vol. 87, no. 3, 2026, doi: 10.32604/cmc.2026.077388.

[13] C. Dupuy, J. Majmudar, J. Wang, T. G. Roosta, R. Gupta, C. Chung, J. Ding, and S. Avestimehr, "Quantifying catastrophic forgetting in continual federated learning," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2023, pp. 1–5.

[14] Z. Wang, F. Wu, F. Yu, Y. Zhou, J. Hu, and G. Min, "Federated continual learning for edge-AI: A comprehensive survey," *arXiv preprint arXiv:2411.13740*, 2024.

[15] Z. Jin, J. Zhou, B. Li, X. Wu, and C. Duan, "FL-IIDS: A novel federated learning-based incremental intrusion detection system," *Future Gener. Comput. Syst.*, vol. 151, pp. 57–70, 2024.

[16] Z. Tan, Q. Wang, C. Anagnostopoulos, S. Puthiya Parambath, J. Singer, and S. Temple, "Distributed log-driven anomaly detection system based on evolving decision making," in *Proc. IEEE 45th Int. Conf. Distrib. Comput. Syst. Workshops (ICDCSW)*, 2025, pp. 231–236, doi: 10.1109/ICDCSW63273.2025.00045.

[17] J. Kirkpatrick *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proc. Nat. Acad. Sci. (PNAS)*, vol. 114, no. 13, pp. 3521–3526, 2017.

[18] P. He, J. Zhu, Z. Zheng, and M. R. Lyu, "Drain: An online log parsing approach with fixed depth tree," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, 2017, pp. 33–40.

[19] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars, "A continual learning survey: Defying forgetting in classification tasks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 7, pp. 3366–3385, 2022.

[20] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny, "Efficient lifelong learning with A-GEM," in *Proc. 7th Int. Conf. Learn. Represent. (ICLR)*, 2019.

VANSH MUNDHRA Vansh Mundhra is a final-year B.Tech. student specializing in Computer Science and Engineering (Core CSE) at Vellore Institute of Technology (VIT), Chennai. He has published research papers in various national

and international conferences. His research interests include Machine Learning, Artificial Intelligence, and Cybersecurity. He is passionate about exploring emerging technologies and applying intelligent computational techniques to address real-world challenges.

SANKARI M She is currently working as an Assistant Professor in the School of Computer Science and Engineering (SCOPE) at Vellore Institute of Technology (VIT), Chennai. She completed her Ph.D. in 2021 from Hindustan Institute of Technology and Science (HITS), Chennai. She received her Master of Engineering (M.E.) in Computer Science and Engineering from Anna University, Chennai, in 2009 and her Bachelor of Engineering (B.E.) in Computer Science and Engineering from Anna University, Chennai, in 2006.

She has over seven years of teaching experience and six years of research experience. She has published more than 20 national and international journal and conference papers in reputed publications indexed in Scopus and Web of Science, including publications by Springer and IEEE. She received the Best Researcher Award in 2022 and the Best Mentor Performer Award in 2023 from NPTEL-SWAYAM. She has also participated in and organized more than 25 Faculty Development Programmes (FDPs), workshops, and research-oriented seminars. Her areas of research interest include Image Security, Deep Learning, Privacy, Image Processing, Artificial Intelligence, and Cloud Computing.

PINAK DEBNATH Pinak Debnath is a final-year B.Tech. student specializing in Computer Science and Engineering (Core CSE) at Vellore Institute of Technology (VIT), Chennai. He has published research papers in various national and international conferences. His research interests include Machine Learning, Artificial Intelligence, and Cybersecurity. He is passionate about exploring emerging technologies and applying intelligent computational techniques to address real-world challenges.

Author contributions Author 1: Conceptualization, Methodology, Software, Formal Analysis, Investigation, Data Curation, Writing – Original Draft. Author 2: Validation, Supervision, Writing – Review & Editing, Project Administration. Author 3: Resources, Visualization, Data Curation, Writing – Review & Editing. All authors reviewed and approved the final version of the manuscript.

Funding There was no specific funding.] **Competing interests** The authors declare no competing interests.

Additional information Supplementary information accompanies this paper. Correspondence and requests for materials should be addressed to Sankari M.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- ScientificReportsSupplementaryInformation1.docx