Amnesia: A Stealthy Replay Attack on Continual Learning Dreams

Anonymous authors
Paper under double-blind review

Abstract

Continual learning (CL) models rely on experience replay to mitigate catastrophic forgetting, yet their robustness to replay sampling interference is largely unexplored. Existing CL attacks mostly modify inputs or update pipelines (poisoning/backdoors) and lack explicit *auditable* constraints, limiting their realism. Here, *auditability* means that a monitor can verify compliance using sampler-visible telemetry, e.g., logged replay index/label statistics, by checking that the realized replay class histogram stays close to a nominal baseline and that the replay rate is unchanged (per-batch and/or over a rolling window). We study a limited-privilege insider controlling only the replay *index selection*, not pixels, labels, or model parameters, while staying within such auditable limits (e.g., queue priorities). We introduce **Amnesia**, a replay composition attack maximizing model degradation under two auditable budgets: a visibility budget δ bounding the TV/KL divergence from a nominal class histogram p_0, and a mass budget f fixing the replay rate. Amnesia uses a two-step procedure: (i) compute lightweight class utilities (e.g., EMA loss/confidence) to tilt p_0 toward harmful classes; (ii) project the tilt back into the δ-ball using efficient KL (*exponential tilt*) or TV (*balanced mass redistribution*) optimizers. A windowed scheduler enforces rolling audits. Across challenging CL benchmarks (Split CIFAR-10/100, CORe50, Tiny-ImageNet) and strong replay baselines (ER, ER-ACE, SCR, DER++), Amnesia consistently depresses final accuracy (ACC↓) and worsens backward transfer (−BWT ↑). The KL variant achieves high impact while remaining largely undetected by audits, as confirmed empirically under multiple audit schemes (per-batch and rolling-window checks), whereas the TV variant is more damaging but more easily detected, especially under tight per-class constraints. These results expose *index-only* replay control as a practical, auditable threat surface in CL systems and establish a principled impact-visibility-budget trade-off. Code is available anonymously via Anonymous GitHub.

1 Introduction

Continual learning (CL) aims to sequentially adapt to evolving data while avoiding catastrophic forgetting (Chaudhry et al., 2019a; Kirkpatrick et al., 2017). Managing the plasticity-stability trade-off (Lange et al., 2021) underpins improved forward and backward transfer (FWT/BWT) (Lopez-Paz & Ranzato, 2017; Lange et al., 2021). This capability is crucial for non-stationary applications such as robotics (Ye & Bors, 2025) and spans task, class, and data-incremental regimes (van de Ven et al., 2022) (e.g., distinct tasks, expanding label sets, or distribution shift under a fixed label set). Replay methods interleave a small memory of past examples with current data to mitigate forgetting (van de Ven et al., 2022), often outperforming regularization alternatives (Chaudhry et al., 2019b; Lopez-Paz & Ranzato, 2017) and proving effective in reinforcement learning (Schaul et al., 2016). Yet the *robustness* of CL pipelines to malicious interference remains underexplored.

In deployed CL systems, replay is often implemented as a sampler/data-service that is instrumented for reliability and compliance: it logs replay index sets, per-batch label counts, queue/priority metadata (e.g., in prioritized replay), and the replay rate (resource usage). Such signals enable lightweight *audits* that verify simple invariants without inspecting model parameters or the live stream, e.g., that the replay keep

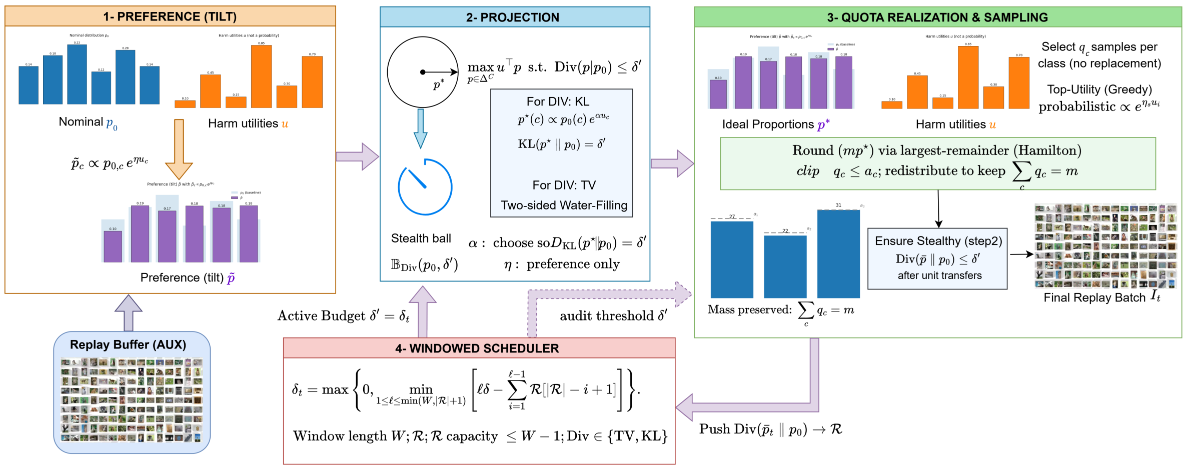

Figure 1: **Amnesia attack overview.** Four-stage pipeline: (1) *Preference*: tilt the nominal class histogram p_0 using harm utilities u to obtain $\tilde{p}$. (2) *Projection*: map into the stealth (divergence) ball $\mathbb{B}_{\mathrm{Div}}(p_0, \delta')$ (total variation / Kullback-Leibler; TV/KL) to get $p^\star$. (3) *Quota & sampling*: round $mp^\star$ to integer quotas q, clip/audit to keep $\mathrm{Div}(\bar{p}\|p_0) \leq \delta'$, then sample the batch indices I_t. (4) *Windowed scheduler*: a ring buffer $\mathcal{R}$ sets the active budget over a rolling window W.

fraction stays fixed and that the replay label histogram remains close to a nominal baseline derived from the buffer. We call constraints of this form *auditable budgets* because they are (i) defined on telemetry that is naturally logged in production pipelines and (ii) verifiable post hoc (per batch and/or over rolling windows) via automated compliance checks. We address the above robustness gap by formalizing these auditable budgets and proposing a *sampler-level*, divergence-constrained replay composition attack operating under explicit visibility and mass limits—to the best of our knowledge, the first to target replay *index selection* under such audit-aligned constraints.

A growing body of work attacks CL via *poisoning* (distribution perturbations) and *backdoors* (input-space triggers). Poisoning aims to degrade overall performance, whereas backdoors implant dormant triggers for targeted failures. For example, *BrainWash* shows that poisoning the *current* task can erase knowledge of *past* tasks (Abbasi et al., 2024); biased synthetic samples can subvert generative replay (Kang et al., 2023); and task-specific poisoning degrades regularization-based CL by exploiting stability assumptions (Han et al., 2023). Backdoor variants such as *PACOL* (Umer et al., 2023) and persistent attacks across task sequences (Guo et al., 2025a) leverage low-intensity triggers or temporally embedded patterns. However, these methods typically omit *practical* constraints that are central in monitored deployments (Steinhardt et al., 2017; Jagielski et al., 2018): (i) an *auditable visibility budget* bounding divergence (δ) between the attacked replay histogram and a nominal baseline p_0 (as checked from selection logs), and (ii) a *mass budget* (f) limiting the replay rate (as checked from resource/throughput accounting). Such budgets are key for realistic, stealthy attacks (Namkoong & Duchi, 2016; Sinha et al., 2018).

We introduce **Amnesia**, a *divergence-constrained, sampler-level* replay composition attack (overview in Fig. 1). Rather than corrupting data, losses, or parameters, Amnesia biases *which indices* are drawn from the buffer (the "dreaming" stage), reflecting practical threats (e.g., a compromised index-selection service, flipped metadata priority flags, or tampered pseudorandom number generator (PRNG) seeds for prioritized replay (Schaul et al., 2016)). Importantly, the sampler need not query the model for real-time per-index losses: the utility signals it consumes can be lightweight, lagged telemetry (e.g., EMA loss/confidence) exported asynchronously by the training service and logged as buffer metadata. We pose the attack as:

$$\max_{p \in \Delta^C} \langle u, p \rangle \quad \text{s.t.} \quad \mathrm{Div}(p\|p_0) \leq \delta, \quad \mathrm{Div} \in \{\mathrm{TV}, \mathrm{KL}\}, \tag{1}$$

where C is the number of classes, u is the attacker's per-class harm utility, and δ is the maximum allowed total variation (TV) distance or Kullback-Leibler (KL) divergence from p_0 (the visibility budget). This is coupled with a mass budget: a keep fraction f that fixes the total replay mass $m = \lfloor f\, n_{\text{aux}} \rfloor$, where n_{aux} is the size of the auxiliary replay buffer. The optimizer's solution, $p^\star$, is then realized as integer quotas q (with $\sum_c q_c = m$); the resulting normalized histogram $\bar{p} = q/m$ is what must pass the audit. The optimizer is agnostic to how u is estimated (e.g., class-wise exponential moving average (EMA) losses, misclassification sensitivity, or forgetting proxies). *Remark:* if u is uninformative (e.g., constant across classes) or $\delta = 0$, then $p^\star = p_0$ and the attack reduces to the nominal replay sampler.

This formulation yields efficient, exact optimizers: a KL *single-tilt* (exponential tilt with a one-dimensional search for the budget-saturating multiplier) and a TV *two-sided water-filling* (balanced mass redistribution between low- and high-utility classes). The attack is designed to maximize forgetting on past tasks while preserving current-task accuracy for stealth, and we evaluate it along three axes: *impact* (backward transfer, BWT; forgetting, FGT (average accuracy drop on past tasks)), *visibility* (δ), and *budget* (f). Under fixed replay mass m, shifting replay probability toward attacker-designated classes necessarily reduces rehearsal for other classes, which is the basic mechanism by which forgetting can be amplified.

Conceptually, Amnesia tilts sampling toward attacker-designated harmful classes while enforcing δ and f. The nominal histogram p_0 may be the instantaneous empirical distribution or a moving average, enabling audits per batch or over a rolling window W. A *windowed scheduler* based on a ring buffer $\mathcal{R}$ tightens the per-step budget to an active $\delta_t \leq \delta$ so that any window of length $L \leq W$ stays within the global radius while meeting integer quotas. This preference-under-constraints view connects replay sampling to our baselines, **PO** (Preference-Only tilt) and **PrO** (Projection-Only fairness/coverage via KL/TV projection around p_0)—but *inverts the goal under the same audit-aligned constraints*: rather than *defending* by minimizing worst-case risk within a divergence ball or enforcing coverage, we *attack* by maximizing targeted harm while remaining compliant with logging-based replay audits (rate and histogram checks). By construction, our optimizer *weakly dominates* any other feasible sampler-level attack (including simple budget-aware greedy heuristics) at an equivalent (f, δ), and *strictly dominates* when u is non-uniform and $\delta > 0$. In summary, our contributions are as follows:

- **Amnesia:** a practical *sampler-level* replay composition attack that steers class proportions by manipulating replay index selection, leaving pixels, loss computations, and model parameters untouched.
- **Principled formulation:** (i) *Attack formulation under auditable budgets:* an explicit divergence-constrained optimization over replay proportions $p \in \Delta^C$ with a *visibility budget* $\mathrm{Div}(p \| p_0) \leq \delta$ (TV/KL distance from the nominal replay histogram) and a *mass budget* f fixing the replay rate $m = \lfloor f\, n_{\text{aux}} \rfloor$; (ii) *Audit realization on what monitors can check:* a sampler-side realization pipeline that converts $p^\star$ into integer quotas q, handles availability constraints, and performs post-rounding *audit-and-fix* so the *realized* replay histogram $\bar{p} = q/m$ satisfies $\mathrm{Div}(\bar{p} \| p_0) \leq \delta'$, with a windowed scheduler enforcing rolling-window audits.
- **Comprehensive evaluation:** experiments on Split CIFAR-10/100, CORe50, and Tiny-ImageNet, against strong replay baselines (ER, ER-ACE, SCR, DER++), systematically analyzing the *impact–visibility–budget* trade-off, damage-per-budget, and attack detectability under multiple audit schemes.

2 Related Work

In continual learning (CL), models learn from streams of tasks or shifting distributions while mitigating catastrophic forgetting (van de Ven et al., 2022; Kirkpatrick et al., 2017). As reviewed in §1, canonical settings are *task-incremental* (task identity known at test), *domain-incremental* (also known as (a.k.a.) data-incremental in some works; shared label space with distribution shift), and *class-incremental* (recognition over all seen classes without task labels) (van de Ven et al., 2022; De Lange et al., 2022; Cossu et al., 2022). CL is also studied in *online, single-pass* streams, especially with many small tasks (Chaudhry & others, 2019). From the evolving accuracy matrix, *Average Accuracy (ACC)*, *Backward Transfer (BWT)*, and *Forward Transfer (FWT)* (typically vs. random initialization) capture final performance, forgetting, and positive transfer (Díaz-Rodríguez et al., 2018; Hou et al., 2024). Method families include *memory-based (replay)*, *regularization-based*, and *architectural/parameter isolation*, with replay dominant in vision; small buffers curb forgetting and are maintained via *reservoir sampling* or *ring-buffer* updates (De Lange et al., 2022; Chaudhry & others, 2019; Mai & contributors, 2020). In practice, replay batches are formed by a sampler module (often implementing class-balancing or prioritized replay), which exposes a natural control point for attacks that manipulate *which* buffer indices are rehearsed. Strong baselines such as *GDumb* highlight buffer composition, while distillation-enhanced replay (e.g., *DER/DER++*) stabilizes predictions; regularization (e.g., elastic constraints) and associated analyses illuminate forgetting (Prabhu et al., 2020; Buzzega et al., 2020b; Kirkpatrick et al., 2017; Huszár, 2018; Shen et al., 2023). The objective is high ACC, non-negative BWT, and positive FWT under realistic memory/compute budgets (Díaz-Rodríguez et al., 2018; De Lange et al., 2022).

Sequential training opens attack surfaces beyond static settings: (i) **poisoning**: insert or modify stream samples to degrade retention or bias behavior (targeted erasure (Li & Ditzler, 2022); BrainWash uses norm-bounded current-task perturbations and particularly harms Elastic Weight Consolidation (EWC) (Abbasi et al., 2024; Kirkpatrick et al., 2017; Huszár, 2018)); small budgets suffice via sequential amplification (Guo et al., 2025b), and replay can repeatedly rehearse poisons (Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2019a); (ii) **backdoors (Trojans)** persisting across tasks, including controllable backdoors effective against regularization- and replay-based learners (Gao & Liu, 2025; 2024), and *Persistent Backdoor Attacks*: *Blind Task Backdoor* (BTB, per-task) and *Latent Task Backdoor* (LTB, single-task "sleeper" activated when a future target class appears), achieving high attack success with minimal clean-accuracy drop (Guo et al., 2025b); (iii) **test-time evasion** where standard adversarial examples remain effective, motivating *Continual Adversarial Defense (CAD)* and *Retrospective Adversarial Replay (RAR)* (Wang et al., 2023; Kumari et al., 2022); and (iv) **distribution/scheduling attacks** that adversarially order tasks to exacerbate interference and forgetting, exploiting order sensitivity and the difficulty of the class-incremental regime (van de Ven et al., 2022; De Lange et al., 2022; Cossu et al., 2022). These threats imply that robust CL must handle sequentially amplified poisoning, long-lived backdoors, non-stationary adversaries, and adversarial curricula.

We systematize these threats along three axes: **Attack Budget (AB)**, **Attack Visibility (AV)**, and **Attack Impact (AI)**. *AB*: attacker control over the stream or training pipeline (poisoned fraction, task access, perturbation magnitude); sequential training can amplify small budgets, e.g., targeted erasure (Li & Ditzler, 2022), ∼4% poisoning in backdoors (Guo et al., 2025b), or norm-bounded task-t perturbations in BrainWash (Abbasi et al., 2024). Temporal access matters: *LTB* needs single-task access, whereas *BTB* assumes per-task intervention (Guo et al., 2025b); controllable backdoors in class-incremental CL succeed against both regularization- and replay-based learners (Gao & Liu, 2025; 2024). Replay revisitation can exacerbate small-budget poisoning (Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2019a). *AV*: stealth, often preserving current-task accuracy or using rare triggers; BrainWash contrasts "reckless" vs. "cautious" trade-offs (Abbasi et al., 2024); persistent backdoors aim for high clean accuracy with long-term success (Guo et al., 2025b; Gao & Liu, 2025). *AI*: damage and persistence, backdoors via attack success across subsequent tasks (Guo et al., 2025b; Gao & Liu, 2025; 2024); poisoning-induced forgetting via negative BWT and ACC drops (Díaz-Rodríguez et al., 2018), with certain regularizers being notably vulnerable (Kirkpatrick et al., 2017; Huszár, 2018). Despite their centrality, AB/AV/AI are rarely reported jointly or standardized, motivating unified evaluation.

3 Methodology

We study a *sampler-level composition attack* for continual learning (CL) with experience replay. The adversary controls only the replay *index set* and seeks to maximize forgetting *(measured as the decline in past-task accuracy)*. The attack has two stages: (1) **Preference**: use lightweight utility signals (e.g., loss, confidence, or other logged scores) to *tilt* the nominal class histogram (p_0), producing a harm-biased target mix; and (2) **Projection**: map this mix into the auditable *stealth ball* ($\mathbb{B}_{\mathrm{Div}}(p_0, \delta)$) with $\mathrm{Div} \in \{\mathrm{TV}, \mathrm{KL}\}$. Projection guarantees each replay batch remains within divergence (δ) of (p_0), keeping batch-level telemetry (the quantities visible to audits) plausibly benign. Unlike harm-agnostic baselines (e.g., **PO** (Preference-Only)) or fairness/coverage quota schemes (e.g., **PrO** (Projection-Only)), we *explicitly* insert the harm-based preference *before* projection, optimizing impact while staying within the same auditable constraints (Kumar et al., 2024).

Crucially, the "tilt-then-project" step is *computationally lightweight*: for **TV**, the exact optimizer is a greedy two-sided water-filling procedure (Algorithm 2 (ProjectTV)); for **KL**, it reduces to a single-parameter exponential tilt with a monotone 1-D root search (Algorithm 2 (ProjectKL)). If utilities are uninformative (e.g., u is constant) or if the active budget is $\delta' = 0$, then projection returns $p^\star = p_0$ and the attack has no effect.

3.1 Preliminaries & Notation

We consider a continual learner with a labeled replay buffer $\mathbf{AUX} = \mathcal{A} = \{(x_i, y_i)\}_{i=1}^{n_{\text{aux}}}$. In class-incremental CL, the set of seen classes grows over time; we let C_t denote the number of classes observed up to step t, interpret labels as $y_i \in [C_t]$, and represent all class histograms (e.g., p_0, $p^\star$, $\bar{p}_t$) as elements of Δ^{C_t}. When new classes appear, we conceptually append new coordinates and recompute p_0 from the current buffer histogram; the divergence audit $\mathrm{Div}(\bar{p}_t \| p_0)$ is then evaluated in this expanded space. While our attack applies to evolving buffers (e.g., sliding windows), we assume $\mathcal{A}$ is a fixed reservoir for notation simplicity. The nominal (audited) class histogram is $p_0 \in \Delta^{C_t}$, where $\Delta^{C_t} = \{p \in \mathbb{R}_+^{C_t} : \sum_c p_c = 1\}$ is the probability simplex; we assume $p_{0,c} > 0$ for all c (or use standard smoothing) so that $\mathrm{KL}(p\|p_0)$ is finite whenever $p_c > 0$. For brevity, when the step t is clear we sometimes write C as shorthand for the current C_t.

At each training step t, the attacker must select a replay batch of a fixed *mass* (size) $m = \lfloor f\, n_{\text{aux}} \rfloor$, determined by a public keep fraction f. This selection is represented by an index set I_t, which corresponds to integer per-class quotas $q_t \in \mathbb{N}_0^C$ (with $\mathbb{N}_0$ for non-negative integers) where $\sum_c (q_t)_c = m$, and realizes the per-batch histogram $\bar{p}_t := q_t/m$. Let a_c be the available samples per class; quotas are clipped ($q_{t,c} \le a_c$) and redistributed to preserve $\sum_c q_{t,c} = m$.

Harm utilities and how they are obtained. The attacker's objective is to maximize harm, quantified by per-class utilities $u = (u_1, \ldots, u_C) \in \mathbb{R}^C$. Optional per-sample utilities $\tilde{u}_i$ (e.g., loss/confidence) are logged as *scalar metadata* alongside AUX items, and class utilities are maintained as EMAs, $u_c^{(t)} = \rho\, u_c^{(t-1)} + (1 - \rho)\, \mathrm{Agg}\{\tilde{u}_i : y_i = c\}$. Importantly, the sampler need not query the *current* model online: $\tilde{u}_i$ can be produced by the training service when an AUX sample is replayed (then stored), refreshed periodically (batch jobs), or replaced by simpler proxies (e.g., age, misclassification counters). If these signals are stale/noisy, the attack degrades gracefully; if u is effectively constant, the optimizer returns $p^\star = p_0$.

We work with divergences $\mathrm{Div} \in \{\mathrm{TV}, \mathrm{KL}\}$, defined as $\mathrm{TV}(p\|p_0) = \frac{1}{2}\|p - p_0\|_1$ and $\mathrm{KL}(p\|p_0) = \sum_c p_c \log\big(\frac{p_c}{p_{0,c}}\big)$. A *stealth radius* $\delta > 0$ defines the *stealth ball* $\mathbb{B}_{\mathrm{Div}}(p_0, \delta) := \{p \in \Delta^C : \mathrm{Div}(p\|p_0) \le \delta\}$. For systems with windowed auditing (length W), the window-average histogram is $\hat{p}_{t-L+1:t} := \frac{1}{L}\sum_{s=t-L+1}^{t} \bar{p}_s$. A ring buffer $\mathcal{R}$ tracks past divergences, and the attacker computes an *online tightened budget* $\delta_t \le \delta$; we use δ' to denote the active budget (either δ or δ_t) at step t. Throughout, we treat $p \in \Delta^C$ as a probability vector (sum $= 1$); the keep fraction f appears only via the batch mass $m = \lfloor f\, n_{\text{aux}} \rfloor$ when realizing $p^\star$ as integer quotas, and all divergences are evaluated on normalized histograms (e.g., $\bar{p}_t = q_t/m$). Finally, class-level preference strength is denoted by $\eta > 0$ and within-class (intra-class) selection by a temperature $\eta_s > 0$.

3.2 Threat Model & Attack Surface

We consider a **grey-box** insider who controls only the replay sampler and aims to maximize forgetting under fixed replay mass f and an audited stealth radius δ; Table 1 summarizes capabilities and limitations.

We assume the auditor's nominal baseline p_0 is computed as a deterministic function of sampler-visible telemetry (e.g., the replay-buffer label histogram at sampling time, or a moving average thereof), so it can be reproduced by the sampler. If the attacker instead observes only a lagged/noisy estimate $\hat{p}_0$, baseline mismatch effectively reduces the usable visibility margin; Appendix B.2 (Table 9) quantifies the resulting impact–stealth trade-off by varying the lag between $\hat{p}_0$ and p_0.

Scope. Our threat model assumes *replay-based* CL with a stored buffer and a distinct replay sampler that selects indices I_t. Accordingly, Amnesia does *not* directly apply to *rehearsal-free* CL methods (e.g., prompt-based approaches such as OVOR Huang et al. (2024) or CODA-Prompt Smith et al. (2023)) because they do not maintain a replay buffer or replay-index selection surface. Conversely, the attack surface is orthogonal to whether the learner fine-tunes the full backbone or updates only a parameter-efficient module (e.g., head/adapters/LoRA/prompts): whenever replay is used, changing the sampled replay indices changes the training signal seen by whatever parameters are trainable. Extensions to other CL pipelines that sample "past information" (e.g., prototype selection or class-conditional generative replay) are conceptually related but outside our experimental scope. More broadly, this suggests potential relevance beyond replay-based CL: future non-replay settings, including continual learning for language models, may expose analogous low-privilege attack surfaces wherever a mechanism selects or composes past information (e.g., retrieval/memory selection, prompt/example selection, or routing), even without an explicit replay buffer; we leave direct study of such settings to future work.

Table 1: Threat model: insider control of the replay sampler with constrained budgets and audited visibility.

Capabilities	**Attacker Limitations**
Direct control of the replay sampler: set per-class quotas/probabilities; realize index sets I_t with total mass m.	No access to model internals (weights, gradients, optimizer).
Read-only visibility of AUX-buffer labels and *stored* utility scores ($\tilde{u}_i$, class-wise u_c); signals may be lagged EMAs or periodically refreshed logs.	No read or write access to the live task stream (pixels, labels) and no requirement to query real-time per-index losses from the trainer (utilities are treated as logged metadata).
Temporal planning: schedule compositions and precommit plans for the next N batches.	Audited controls: fixed keep fraction f and stealth radius δ; TV/KL checks on label histograms vs. baseline p_0.
	Active monitoring: auditors track per-batch/window histograms/divergences and cross-check sampler code/config and selection logs.

This attack surface is realistic in modern MLOps pipelines. Integration points include sampler plugins that intercept batch requests (e.g., `get_replay_batch()`), configuration toggles for "balanced replay" that route to a quota module, and data-ops jobs that precompute a selection plan for future windows. Legitimate controls such as *class-balancing*, *prioritized replay*, and *curriculum sampling* are commonly exposed via standard **PyTorch** sampler/data-loader hooks and the same utility telemetry, making this insider surface viable. The attacker can either persistently write malicious index sets or have the sampler execute a precomputed plan.

3.3 Preference–Projected Replay Attack

This subsection details the sampler-side routine as shown in Fig. 1. It runs once per training step, manipulates only replay *indices*, and leaves model weights/gradients/loss unchanged. At each step we solve the class-level program

$$\max_{p\in\Delta^C} \; u^\top p \quad \text{s.t.} \quad \mathrm{Div}(p\|p_0) \le \delta', \quad \mathrm{Div} \in \{\mathrm{TV}, \mathrm{KL}\}, \tag{2}$$

where u are harm utilities and δ' is the active stealth budget (either the static δ or the dynamic windowed budget δ_t).

Step A: Preference (tilt). Encode harm preference using per-class utilities u_c and a tilt strength $\eta > 0$. Define the (unconstrained) tilted mix $\tilde{p}_c \propto p_{0,c}\, e^{\eta u_c}$. For **TV** divergence, only the *ordering* of u_c matters in the next step; the **KL** solution is determined exactly by the projector (Step B). When u is constant, tilting has no effect and $\tilde{p} = p_0$.

Algorithm 1 Amnesia Replay Attack

Require: AUX buffer $\mathcal{A} = \{(x_i, y_i, \tilde{u}_i)\}_{i=1}^{n_{\text{aux}}}$; nominal histogram $p_0 \in \Delta^C$; keep fraction f; stealth radius δ; window W; divergence Div $\in \{\text{KL}, \text{TV}\}$; class utilities $u_{1:C}$; sample temperature $\eta_s > 0$; ring buffer $\mathcal{R}$ of past divergences (size $\leq W - 1$)

Ensure: Replay indices I with exact mass m and enforced per-batch/window stealth

1: $m \leftarrow \lfloor f \cdot n_{\text{aux}} \rfloor$
2: **if** $W > 1$ **then**
3: $\quad \delta' \leftarrow \max\Big\{0, \min_{0 \leq L \leq \min(W-1, |\mathcal{R}|)} \Big((L+1)\delta - \sum_{\ell=|\mathcal{R}|-L+1}^{|\mathcal{R}|} \mathcal{R}[\ell]\Big)\Big\}$ ▷ empty sum = 0 for $L = 0$
4: **else**
5: $\quad \delta' \leftarrow \delta$
6: **end if**
7: **if** Div = KL **then**
8: $\quad p^\star \leftarrow$ PROJECTKL(p_0, u, δ') ▷ Algorithm 2 (ProjectKL)
9: **else** ▷ Div = TV
10: $\quad p^\star \leftarrow$ PROJECTTV(p_0, u, δ') ▷ Algorithm 2 (ProjectTV)
11: **end if**
12: $q \leftarrow$ ROUNDTOSUM$(m\, p^\star)$
13: $q \leftarrow$ CLIPTOAVAILABILITY$(q, \mathcal{A})$
14: $q \leftarrow$ AUDITANDFIXQUOTAS$(q, p_0, \delta', \text{Div})$ ▷ unit transfers until $\text{Div}(q/m \| p_0) \leq \delta'$
15: $\bar{p} \leftarrow q/m$
16: $I \leftarrow \emptyset$
17: **for** each class c **do**
18: $\quad$ add $q[c]$ indices of class c by top-$\tilde{u}_i$ or by sampling $\propto \exp(\eta_s \tilde{u}_i)$ without replacement
19: **end for**
20: push $\text{Div}(\bar{p} \| p_0)$ to $\mathcal{R}$
21: **if** $|\mathcal{R}| > W - 1$ **then**
22: $\quad$ pop oldest
23: **end if**
24: **return** I

Step B: Projection to the stealth ball (exact optimizers). We find the optimal proportions $p^\star$ that solve Eq. 2, attaining the maximum harm while satisfying $\text{Div}(p^\star \| p_0) \leq \delta'$. The method differs for KL and TV:

- **KL ball (single-tilt search).** By Lagrangian optimality, the optimizer has exponential form

$$p^\star(c) \;=\; \frac{p_0(c)\, \exp(\alpha\, u_c)}{\sum_j p_0(j)\, \exp(\alpha\, u_j)}, \tag{3}$$

  where a scalar $\alpha \geq 0$ is chosen so that $\text{KL}(p^\star \| p_0) = \delta'$ whenever the constraint is active; $\text{KL}(p_\alpha \| p_0)$ is monotone in α when u is non-constant, so a safe bisection finds α (Algorithm 2 (ProjectKL)). Each evaluation is $O(C)$; overall cost is $O(C \cdot \text{iters})$.

- **TV ball (two-sided water-filling).** Sort classes by u_c and greedily move probability mass from the *lowest*-utility classes to the *highest*-utility classes until the total ℓ_1 budget $\sum_c |p_c - p_{0,c}| = 2\delta'$ is exhausted, respecting simplex bounds (Algorithm 2 (ProjectTV)). This is an exact solution and is inherently "greedy constraint" in nature: it spends the limited TV budget only on transfers that maximally increase $u^\top p$.

Both methods return the exact optimizer $p^\star$ of Eq. 2.

Step C: Budgeted quotas and within-class selection.

Convert $p^\star$ to practical integer quotas $q \in \mathbb{N}_0^{C_t}$ (Alg. 1, Lines 9–11): (i) apply *largest-remainder rounding* (Hamilton method (Janson & Linusson, 2012)): set $q_c \leftarrow \lfloor m\, p_c^\star \rfloor$, then allocate the remaining $m - \sum_c q_c$ units to classes with the largest fractional parts of $m\, p_c^\star$ until $\sum_c q_c = m$; (ii) *clip to availability* ($q_c \leq a_c$) and redistribute any deficit while preserving the total mass m; (iii) run *audit-and-fix* to ensure $\text{Div}(q/m \| p_0) \leq \delta'$.

Algorithm 2 Projection Solvers for KL and TV Balls

ProjectKL (left panel): PROJECTKL(p_0, u, δ')

```
function TILT(α):
    p_α ← p_0 exp(αu); return p_α / Σ p_α
if max u = min u then return p_0 end if
α_lo ← 0; α_hi ← 1
while KL(TILT(α_hi) ‖ p_0) < δ' do
    α_hi ← 2 α_hi
end while
while α_hi − α_lo > ε_α do
    α ← (α_lo + α_hi)/2; p_α ← TILT(α)
    if KL(p_α ‖ p_0) < δ' then
        α_lo ← α
    else
        α_hi ← α
    end if
end while
return p* ← TILT(α_hi)
```

ProjectTV (right panel): PROJECTTV(p_0, u, δ')

```
p ← p_0; b ← 2δ'
Sort indices: u_(1) ≤ ··· ≤ u_(C)
ℓ ← 1 (donor), r ← C (receiver)
while b > 0 and ℓ < r do
    ε ← min{ p_(ℓ), 1 − p_(r), b/2 }
    p_(ℓ) ← p_(ℓ) − ε
    p_(r) ← p_(r) + ε
    b ← b − 2ε
    if p_(ℓ) = 0 then ℓ ← ℓ + 1
    end if
    if p_(r) = 1 then r ← r − 1
    end if
end while
return p* ← p
```

Rounding introduces only tiny, explicitly bounded distortions: $\|\bar{p} - p^\star\|_\infty \le 1/m$ and $\|\bar{p} - p^\star\|_1 \le C_t/m$ (i.e., the bound scales with the current number of seen classes).

For **TV**, each single-unit transfer from any class with $\bar{p}_i > p_{0,i}$ to any class with $\bar{p}_j < p_{0,j}$ decreases TV by exactly $1/m$, so feasibility is reached in finitely many swaps (at most $2\delta' m$ swaps in the worst case). For **KL**, we perform discrete "steepest-decrease" swaps (from the largest log-ratio $\log(\bar{p}_c/p_{0,c})$ to the smallest), which strictly decreases KL each step until $\mathrm{KL}(\bar{p}\|p_0) \le \delta'$. Within each class, select indices by *top-*$\tilde{u}_i$ or *probabilistically* with $\Pr(i \,|\, y_i = c) \propto e^{\eta_s \tilde{u}_i}$ (no replacement).

Step D: Windowed visibility (online scheduler). With windowed auditing (length W), we compute a tightened δ_t from the ring buffer $\mathcal{R}$ (Alg. 1, Lines 2–6) and enforce $\mathrm{Div}(\bar{p}_t\|p_0) \le \delta'_t$ at each step. Intuitively, the scheduler spends only the *residual* budget left after accounting for the last $W-1$ steps. In ideal arithmetic, this implies a deterministic sliding-window guarantee:

Proposition (Residual-budget window compliance). Let δ'_t be chosen as in Alg. 1 and enforce $\mathrm{Div}(\bar{p}_t\|p_0) \le \delta'_t$ at every step. If Div is convex in its first argument, then for all t and all $1 \le L \le W$,

$$\mathrm{Div}\big(\hat{p}_{t-L+1:t} \;\|\; p_0\big) \;\le\; \delta. \tag{4}$$

Proof sketch. The residual update ensures the partial sums over any trailing window of length $L \le W$ never exceed $L\delta$. By Jensen/convexity, $\mathrm{Div}(\hat{p}\|p_0) \le \frac{1}{L}\sum_s \mathrm{Div}(\bar{p}_s\|p_0) \le \delta$.

In practice, discretization/availability can introduce rare numerical slack; we therefore also report empirical window-violation rates in §3.4.

Complexity. Scheduler update is $O(W)$. Projection is $O(C \cdot \text{iters})$ (KL) or $O(C \log C)$ (TV, for sorting). Quotas/auditing are $O(C)$ plus a small number of unit transfers.

3.4 Visibility and Efficiency Guarantees

Our method provides three key guarantees by construction, ensuring stealth and budget compliance; we also state an explicit dominance property.

Guarantee 1: Per-batch Stealth (hard constraint) and tail reporting (policy). By Step C's audit-and-fix, the realized histogram satisfies $\mathrm{Div}(\bar{p}_t\|p_0) \le \delta'$ (hence $\le \delta$). For reporting, we define the *normalized* per-batch divergence and its 95th-percentile summary:

$$r_t := \frac{\mathrm{Div}(\bar{p}_t\|p_0)}{\delta} \in [0,1], \qquad r_{\text{batch@95}} := \mathrm{Quantile}_{0.95}\{r_t\}. \tag{5}$$

We use $r_{\text{batch@95}} \leq 0.05$ as a *calibrated reporting threshold*: it means 95% of batches spend at most 5% of the audit radius, leaving headroom for rounding/availability noise. This is a policy choice for distinguishing "highly stealthy" regimes (not a requirement implied by the definition of the audit ball). Appendix reports the clean-run calibration motivating the 0.05 band.

Guarantee 2: Windowed Stealth (deterministic scheduler) and empirical slack. Under Step D's residual-budget scheduler and convexity of Div, Eq. 4 holds for any window $L \leq W$. We additionally report an empirical **window violation rate** to capture any residual exceedances due to discretization/measurement noise:

$$r_{\text{win}} \;=\; \frac{1}{T}\sum_{t=1}^{T} \mathbb{1}\big\{\mathrm{Div}\big(\hat{p}_{t-W+1:t} \,\|\, p_0\big) > \delta\big\}\,. \tag{6}$$

We treat $r_{\text{win}} \leq 0.05$ as an *operational acceptance band* (clean-calibrated) rather than a theoretical necessity.

Guarantee 3: Budget Conservation. The quota generation process (Step C) is strictly mass-preserving, ensuring $\sum_{c=1}^{C} q_{t,c} = m$. This yields a realized keep fraction $\hat{f}_t = m/n_{\text{aux}}$ that tightly tracks the target f, with $|\hat{f}_t - f| \leq 1/n_{\text{aux}}$. We verify this using the **95th-percentile fraction error**:

$$e_{95} \;=\; \mathrm{Quantile}_{0.95}\big(|\hat{f}_t - f|\big). \tag{7}$$

Because mass is exact, we have the deterministic bound $e_{95} \leq 1/n_{\text{aux}}$ (e.g., ≤ 0.002 for n_{aux}=500); we use $e_{95} \leq 0.02$ only as a loose reporting range.

Dominance property (optimality under the audited constraint). Fix any non-constant utility vector u, nominal histogram p_0, divergence type Div $\in \{\mathrm{KL}, \mathrm{TV}\}$, and active budget δ'. Let $p^\star$ be the solution of Eq. 2. Then for any other feasible sampler-level class mix $p \in \Delta^C$ with $\mathrm{Div}(p\|p_0) \leq \delta'$, we have $u^\top p \leq u^\top p^\star$, with strict inequality unless p is also optimal (and in particular if $\delta' > 0$ and u is non-constant, the optimizer is unique for KL). Thus, given a specified harm surrogate u and the same auditable constraint set, Amnesia's projection step is not a heuristic: it is the exact best-response class composition.

4 Experiment Setup

4.1 Datasets

We evaluate on four standard CL benchmarks of increasing difficulty: *Split CIFAR-10* (Zenke et al., 2017) (10 classes, 32×32 RGB; 5 tasks × 2 classes, easy), *CORe50* (Lomonaco & Maltoni, 2017) (50 classes; class-incremental; 10 tasks × 5 classes, harder), *Split CIFAR-100* (Zenke et al., 2017) (100 classes, 32×32 RGB; 10 tasks × 10 classes), and *Tiny-ImageNet* (Stanford, 2015) (ImageNet subset, 200 classes, 64×64; we use 100 classes as 5 tasks × 20 classes and reserve the remaining 100 for auxiliary out-of-stream ablations, more challenging).

4.2 Replay-based Continual Learning Methods

Replay CL maintains a small memory $\mathcal{M}$ and, at step t, optimizes on the union of the current mini-batch B_t and a memory mini-batch $M_t \subset \mathcal{M}$ (typically via reservoir sampling). We attack the canonical **Experience Replay (ER)** (Rolnick et al., 2019) and three prominent extensions: **ER-ACE** (Caccia et al., 2022) (asymmetric cross-entropy restricting logits on new data to current-task classes to reduce representation drift), **DER++** (Buzzega et al., 2020a) (ER with knowledge distillation by matching current logits to stored past logits for memory samples), and **SCR** (Mai et al., 2021) (supervised contrastive loss on mixed new+replay batches to learn a more unified representation).

4.3 Training Protocol

Protocol. Unless stated otherwise, we use **ResNet-18** (He et al., 2016) with **SGD** (lr = 0.03), mini-batch size 64, and buffer size 500. Images follow official splits and are normalized to $[0, 1]$; CORe50 uses the *New*

Table 2: **Amnesia attack results (Impact / Stealth / Budget).** Base (no attack) vs. **KL** and **TV** for all models/datasets. **Impact:** ACC↓, −BWT ↑; **Stealth:** $r_{\text{batch@95}}$ ↓, r_{win} ↓; **Budget:** e_{95} ↓. *Mean±std over 5 seeds. Stealth/Budget are* $\times 10^{-2}$ *(e.g.,* $5 = 0.05$*). Red cells exceed conservative, clean-calibrated reporting bands (*$r_{batch@95} \leq 0.05$*,* $r_{win} \leq 0.05$*,* $e_{95} \leq 0.02$*); these are reporting policies (tail headroom / violation-rate targets), not a change to the enforced per-batch audit constraint* $\text{Div}(\bar{p}\|p_0) \leq \delta'$ *(guaranteed by construction).*

Dataset	Model	No Attack	KL			TV		
			Impact	Stealth	Budget	Impact	Stealth	Budget
		ACC↓ (-BWT↑)	ACC↓ (-BWT↑)	$r_{\text{batch@95}}$ ↓ (r_{win} ↓)	e_{95} ↓	ACC↓ (-BWT↑)	$r_{\text{batch@95}}$ ↓ (r_{win} ↓)	e_{95} ↓
CIFAR-10	ER	50.4±0.6 (55.1±1.3)	29.3±0.29 (82.7±2.0)	5.0±0.4 (0.2±0.1)	1.0±0.3	25.1±0.25 (88.2±2.3)	5.0±2.0 (6.0±1.5)	1.5±0.5
	SCR	59.6±0.7 (38.9±0.4)	**31.1±0.8 (78.2±2.1)**	**4.0±0.3 (0.7±0.2)**	**1.5±0.4**	28.9±0.29 (83.4±2.3)	4.0±0.3 (0.7±0.2)	1.8±0.5
	DER++	64.0±0.7 (29.1±0.3)	33.1±0.9 (69.1±2.0)	5.0±0.5 (0.2±0.1)	0.9±0.3	30.7±0.90 (75.1±2.2)	10±4.0 (1.0±0.5)	2.0±0.7
	ER-ACE	65.2±0.8 (15.2±0.15)	34.2±1.0 (56.9±1.7)	1.0±0.2 (0.1±0.1)	1.0±0.3	32.3±0.95 (60.3±1.8)	4.0±0.4 (3.0±0.8)	2.0±0.7
CORe50	**ER**	55.4±0.7 (52.5±1.5)	32.5±0.8 (84.4±2.1)	0.8±0.2 (0±0.0)	0.9±0.3	**32.1±0.96 (85.8±2.2)**	**4.0±0.3 (0±0.0)**	**0.8±0.3**
	SCR	52.9±0.8 (58.8±1.6)	28.5±0.28 (87.2±2.2)	5.0±0.4 (0±0.0)	0.9±0.3	33.5±1.00 (82.4±2.1)	3.0±0.3 (5.0±0.8)	0.9±0.3
	DER++	61.7±0.8 (47.3±0.47)	30.1±0.90 (86.8±2.2)	8.0±3.0 (0.1±0.5)	1.0±0.6	32.2±0.96 (84.53±2.4)	10±4.0 (0.3±1.0)	1.0±0.8
	ER-ACE	68.7±0.9 (24.5±0.25)	36.0±1.0 (75.7±2.0)	1.4±0.4 (0.1±0.1)	2.0±0.8	34.6±1.00 (74.7±2.1)	4.0±1.5 (3.0±2.0)	0.9±0.6
CIFAR-100	ER	35.2±0.7 (57.4±1.7)	20.3±0.20 (72.7±2.2)	4.0±0.6 (2.0±0.6)	0.3±0.2	21.8±0.22 (73.1±2.2)	5.0±0.7 (3.0±0.6)	1.0±0.5
	SCR	33.5±0.7 (52.3±1.6)	21.5±0.22 (75.6±2.3)	4.0±0.5 (3.0±0.6)	1.2±0.7	19.2±0.19 (81.4±2.4)	15±4.0 (2.0±1.0)	1.9±0.9
	DER++	33.1±0.7 (45.2±0.45)	18.7±0.19 (63.6±1.9)	13±5.0 (2.0±1.0)	1.9±0.9	20.1±0.20 (60.5±1.8)	41±5.0 (6±5.0)	5.0±3.0
	ER-ACE	44.2±0.9 (21.3±0.21)	23.3±0.23 (65.6±2.0)	3.7±0.6 (0±0.0)	0.2±0.2	**21.5±0.22 (70.1±2.1)**	**5.0±0.9 (5.0±0.9)**	**0.9±0.8**
TinyImageNet	ER	17.0±0.17 (21.3±0.21)	8.2±0.08 (39.3±0.39)	4.0±0.6 (3.0±0.6)	0.6±0.5	6.5±0.06 (45.9±0.45)	11±4.0 (12±5.0)	1.2±0.8
	SCR	22.5±0.22 (20.2±0.20)	11.4±0.11 (33.7±0.33)	12±4.0 (2.0±1.0)	1.1±0.8	9.9±0.09 (42.8±0.42)	5.0±0.8 (4.0±0.8)	2.0±0.9
	DER++	21.8±0.22 (19.5±0.20)	10.2±0.10 (30.1±0.30)	11±3.0 (2.0±1.0)	1.2±0.8	11.8±0.12 (38.4±0.38)	34±5.0 (36±5.0)	5.1±3.0
	ER-ACE	23.5±0.24 (15.7±0.16)	**13.1±0.13 (45.1±0.45)**	**2.8±0.6 (1.0±0.6)**	**0.7±0.5**	10.5±0.10 (52.4±1.50)	5.0±0.7 (4.0±0.8)	1.9±0.8

Classes (NC) scenario. Each task is trained for **10 epochs** with a **fixed** task order; results are averaged over **5** seeds. Default replay/audit settings are keep fraction $f = 0.1$, stealth radius $\delta = 0.1$, and audit window $W = 10$.

Clean vs. attacked hyperparameters. For every method/dataset, learner-side hyperparameters are *identical* between clean and attacked runs (optimizer, learning rate, epochs, augmentations, buffer size, and method-specific settings), isolating the effect of *sampler-level replay composition* (index selection) from retuning. Since CL performance can be hyperparameter-sensitive, we focus on *relative* degradation under a fixed, standard protocol rather than globally optimal tuning.

Sampler telemetry used to construct utilities. Amnesia consumes lightweight, *sampler-readable* utility logs produced by the training job and read asynchronously by the sampler. Unless stated otherwise, $\tilde{u}_i$ is the cross-entropy loss of a replayed example when it is last observed during training, and u_c is an EMA over an aggregation (mean) of $\tilde{u}_i$ for samples with label c; thus the sampler requires no real-time per-index model queries and only reads the latest available utility snapshot.

Nominal histogram for auditing. Unless stated otherwise, p_0 is the replay-buffer class histogram at sampling time (with standard smoothing to ensure $p_{0,c} > 0$ for KL), and all audit metrics (TV or KL) are computed between the realized replay histogram and this p_0.

4.4 Evaluation Criteria

Beyond the stealth/budget metrics in §3.4, we report final accuracy and backward transfer (BWT). Let $R_{i,j}$ denote performance on task j after training through task i, and let T be the number of tasks:

$$\text{BWT} = \frac{1}{T-1} \sum_{j=1}^{T-1} \big(R_{T,j} - R_{j,j}\big), \qquad \text{ACC} = \frac{1}{T} \sum_{j=1}^{T} R_{T,j}. \tag{8}$$

By convention, negative BWT indicates forgetting; we therefore report −BWT so that larger values correspond to more forgetting (stronger attack impact).

5 Results

Main impact and audit adherence. Table 2 summarizes *impact* (ACC↓, −BWT ↑) and *audit adherence* (Stealth, Budget). The sampler enforces the core per-batch constraint $\text{Div}(\bar{p}\|p_0) \leq \delta'$ by construction (via audit-and-fix); red denotes exceeding conservative reporting bands: (i) $r_{\text{batch@95}} \leq 0.05$ (95% of steps spend at most 5% of the audit radius; tail headroom) and (ii) $r_{\text{win}} \leq 0.05$ (an operational target for rare window exceedances under discretization/availability noise). Under these criteria, green highlights the best *compliant* entries per setting.

Across datasets and replay methods, Amnesia induces substantial forgetting: ACC drops sharply and −BWT rises. Canonical ER is often (though not always) among the most vulnerable. ER-ACE, which mitigates representation drift, is typically the most robust *among compliant runs* (e.g., Tiny-ImageNet/KL) while still suffering large degradation. DER++ is particularly sensitive in high-class regimes (CIFAR-100, Tiny-ImageNet), exhibiting both strong impact and frequent red-flagged stealth/budget issues. A plausible explanation is that DER++'s distillation loss sharpens the utility landscape and creates more extreme sampling pressures, which are harder to realize cleanly under integer quotas and limited per-class availability.

When and why do red-flagged audit metrics occur? Band exceedances are more common as the *effective granularity* tightens, especially when $m/C \approx 1$ (few items per class per batch). CORe50 (C=50) is representative when f=0.1 (small m), and CIFAR-100 / Tiny-ImageNet are also tight because C is large. In these regimes, discretization (rounding), clipping to availability, and sparse allocations (especially for TV) can create batch-to-batch spikes in normalized divergence and rare window exceedances. This matches §3.3: when m/C is small, even a one-unit quota change yields a large change in $\bar{p}$, so realized histograms become inherently "chunky," increasing the risk of tail spikes in r_t and, for TV, occasional window exceedances under aggressive reallocation.

KL vs. TV trade-off and the "greedy constraint" question. A consistent KL–TV trade-off emerges: TV often yields slightly stronger impact but triggers more red-flagged stealth/budget entries. This follows from the projectors. KL's exponential tilt, $p^\star(c) \propto p_0(c) \exp(\alpha u_c)$, preserves positive mass on all classes and discretizes more smoothly, producing fewer tail/window spikes. TV's two-sided water-filling can push low-utility classes toward 0, producing a sparser and more extreme $p^\star$ (closer to the unconstrained optimum) that is harder to realize under integerization and availability constraints.

This also motivates a simple, budget-aware baseline: *the TV variant is itself greedy and budget-aware.* The TV projector is exactly two-sided water-filling, greedily transferring probability mass from the lowest-utility to the highest-utility classes until the TV budget is exhausted (Algorithm 2 (ProjectTV)). For KL, the "optimization" is a single-parameter monotone search implementing the exponential-tilt solution (Algorithm 2 (ProjectKL)). Thus, the key modeling choice is the harm surrogate u (defined in §3.1 and instantiated in §4.1).

5.1 Robustness to Backbone Choice

Replacing ResNet-18 with ViT-Tiny and ConvNeXt-Base (CIFAR-100, ER-ACE, KL) preserves the qualitative conclusion (Table 3): Amnesia remains strongly effective (large ACC drops and severe increases in forgetting) while staying within the same audit policy (low $r_{\text{batch@95}}$, zero r_{win}, and tight e_{95}). Stronger backbones improve clean performance and show slightly higher robustness under attack (higher attacked ACC and slightly lower forgetting), suggesting that increased representational stability can reduce (but not eliminate) the leverage of divergence-constrained replay steering.

Ablation insights (ER-ACE, Split CIFAR-10). Table 4 disentangles **Preference** (aux_trim) and **Projection** (PrO). *Preference without Projection* (**PO**) can improve clean CL (higher ACC, lower −BWT): PO resembles prioritized replay / hard-example mining and does not enforce an adversarial class-level reallocation under an audit constraint, so it can act as a benign optimization rather than an attack. *Projection without Preference* (**PrO**) increases forgetting but is substantially more visible (large $r_{\text{batch@95}}$ and non-trivial r_{win}), showing that spending divergence budget without harm-targeted preference is inefficient and detectable. The full two-stage method restores the intended harm–stealth balance: **Amnesia (KL)** achieves strong forgetting with clean batch/window compliance, while **Amnesia (TV)** is more aggressive but generally more brittle because sparse, extreme allocations are harder to realize under discretization and availability constraints.

Table 3: **CIFAR-100 (ER-ACE) under (KL attack)**. Values are mean±std over 5 seeds. Stealth and Budget are reported $\times 10^{-2}$ (e.g., 5 denotes 0.05).

Metric	VIT	Convnext
Baseline ACC	48.1±0.8	51.7±0.9
Baseline Forget	20.0±0.20	19.1±0.18
Attack ACC	25.6±0.25	27.9±0.30
Attack Forget	63.8±1.8	61.9±1.7
Stealthy $r_{\text{batch@95}}$	3.5±0.6	3.3±0.5
Stealthy r_{win}	0±0.0	0±0.0
Budget e_{95}	0.2±0.2	0.2±0.2

Table 4: **Ablation of preference and projection on ER-ACE (Split CIFAR-10).** ✓/✗ denote enabled/disabled. **aux_trim** utility prioritization; **PrO** is the projection only **PO**: aux_trim only.

Model	aux_trim	PrO	Div.	ACC (-BWT)	$r_{\text{batch@95}}(r_{\text{win}})$	e_{95}
ER-ACE (Baseline)	✗	✗	N/A	65.21 / 15.25	N/A	N/A
PO	✓	✗	N/A	68.55 / 12.39	N/A	N/A
PrO (KL)	✗	✓	KL	45.07 / 31.67	0.516 (0.224)	0.013
PrO (TV)	✗	✓	TV	47.38 / 26.06	1.183 (0.222)	0.012
Amnesia (TV)	✓	✓	TV	32.34 / 60.30	0.040 (0.085)	0.020
Amnesia (KL)	✓	✓	**KL**	**34.20 / 56.90**	**0.010 (0.000)**	**0.011**

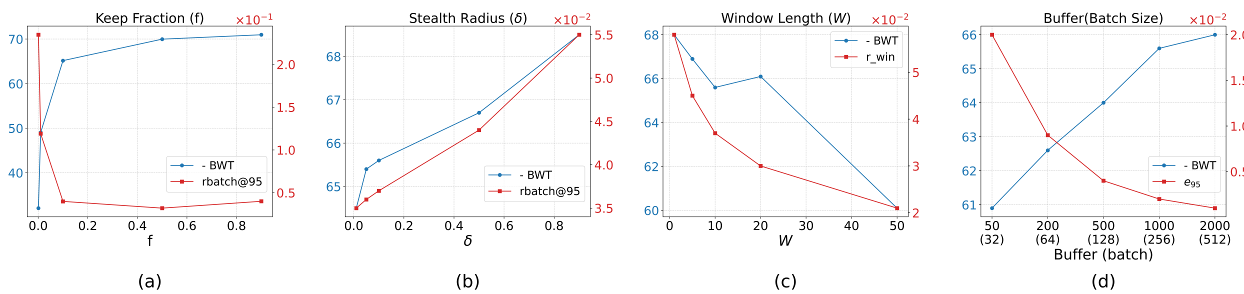

Figure 2: **Ablations (ER-ACE, Split CIFAR-10).** Blue (left axis): $-$BWT (impact). Red (right axis): an audit/budget metric. (a) x: keep fraction f; red: $r_{\text{batch@95}}$ ($\times 10^{-1}$). (b) x: stealth radius δ; red: $r_{\text{batch@95}}$ ($\times 10^{-2}$). (c) x: audit window W; red: r_{win} ($\times 10^{-2}$). (d) x: buffer size (batch size in parentheses); red: e_{95} ($\times 10^{-2}$).

Fig. 2(a,b) show keep fraction f and stealth radius δ effects (blue: higher $-$BWT $\Rightarrow$ more forgetting; red: batch-level audit pressure). In Fig. 2(a), increasing f enlarges replay mass $m{=}\lfloor f n_{\text{aux}} \rfloor$, letting the sampler realize $p^\star$ more faithfully and select more harmful items; thus $-$BWT rises, while $r_{\text{batch@95}}$ falls due to reduced discretization pressure, matching Step C's bound $\|\bar{p}{-}p^\star\|_1 \le C/m$ (larger $m \Rightarrow$ less "chunkiness"). In Fig. 2(b), increasing δ expands the feasible ball, so the optimizer spends more divergence and both $-$BWT and $r_{\text{batch@95}}$ increase monotonically (harm–stealth trade-off).

Fig. 2(c) shows larger W makes the residual-budget scheduler more conservative (smaller δ_t), slightly reducing $-$BWT while driving window violation rate down, consistent with the window-compliance guarantee in §3.4. Fig. 2(d) shows larger buffers strengthen the attack (more replay candidates and stronger within-class choice) while improving budget tracking, since larger buffers/batches reduce integerization effects and availability-driven failure modes.

Because prior baselines do not enforce auditable stealth/budget constraints, Table 5 reports only shared metrics (ACC, $-$BWT) and should be read as *impact-at-any-cost* references. In task-incremental CIFAR-10 with ER, we compare Amnesia with three data-poisoning baselines; Table 5 lists their ACC and $-$BWT. Despite operating under explicit audit-aware constraints and sampler-only access (pixels/labels/parameters untouched), Amnesia achieves strong forgetting, highlighting index-only replay control as a practical threat surface in monitored deployments.

Runtime overhead. Relative to baseline, Amnesia adds modest end-to-end cost: +24 s (≈3.5%) for KL and +49 s (≈7.1%) for TV. This matches sampler-side costs: $O(W)$ scheduling, $O(C)$ quotas/audit, and $O(C)$ (KL) vs. $O(C \log C)$ (TV) projection, which remain small relative to forward/backward passes and support the practicality of sampler-only interference.

Table 5: Comparison between Amnesia and similar attacks.

Attack Method	Result
Targeted Poisoning (Li & Ditzler, 2022)	19.6 (66.3)
PACOL (Umer et al., 2023)	15.8 (68.7)
BrainWash (Abbasi et al., 2024)	25.5 (75.4)
Ours	**29.3 (82.7)**

Table 6: End-to-end training time on CIFAR-10 (mean±std).

Method	Time (min)
Baseline (no attack)	11:34 ± 0:28
Amnesia (KL)	11:58 ± 0:49
Amnesia (TV)	12:23 ± 0:58

Appendix B.3 (Table 10) provides a component-level breakdown of this overhead into (i) projection, (ii) Rounding/Clipping, and (iii) Audit-and-Fix, highlighting that TV is dominated by sorting in projection and by swap counts in audit-and-fix.

6 Conclusion

We identified a realistic, auditable vulnerability in continual learning: *sampler-level* control of replay indices. **Amnesia** casts malicious replay composition as a divergence-constrained program with explicit visibility (δ) and mass (f) budgets, combining a harm-driven *preference* step with an exact *projection* onto a TV/KL stealth ball. The resulting optimizers, a KL *single-tilt* and a TV *two-sided water-filling*, together with a windowed scheduler, are efficient, mass-preserving, and optimal within the audited set. Across Split CIFAR-10/100, CORe50, and TinyImageNet, with strong replay baselines (ER, ER-ACE, SCR, DER++), Amnesia reliably reduces accuracy and increases forgetting while satisfying audits in most settings. KL yields near-maximal damage with high compliance; TV achieves higher impact but is brittle when the mass-per-class ratio is tight. Ablations confirm that both *preference* and *projection* are necessary to attain the intended impact–visibility–budget trade-off.

A limitation of this study is that it assumes labeled buffers and a fixed nominal histogram p_0 for auditing; applicability to unlabeled or evolving label spaces remains unexplored. Moreover, when $m/C \approx 1$, discretization and availability constraints can strain stealth, especially for TV. Future directions can develop sampler-aware defenses (attested index selection, cryptographic logging, randomness beacons) and *multi-metric* auditors beyond class histograms (e.g., MMD/CUSUM, gradient telemetry), and extend the framework to generative or unlabeled replay, task-free online CL, and RL/robotics, alongside theory linking (δ, f) to bounds on expected $-$BWT and detectability. More broadly, future work should test whether analogous selection/composition attack surfaces arise in non-replay CL and continual learning for language models (e.g., retrieval/memory selection, prompt/example selection, or routing). Our findings elevate replay index selection to a first-class security primitive, underscoring the need to secure the data path, not just the model, in deployed CL systems.

References

Ali Abbasi, Parsa Nooralinejad, Hamed Pirsiavash, and Soheil Kolouri. BrainWash: A Poisoning Attack to Forget in Continual Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24057–24066, 2024. 2, 4, 13

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020a. Curran Associates Inc. ISBN 9781713829546. 9

Pietro Buzzega, Matteo Boschini, Angelo Porrello, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. In *Advances in Neural Information Processing Systems*, volume 33, 2020b. URL `https://proceedings.neurips.cc/paper/2020/hash/b704ea2c39778f07c617f6b7ce480e9e-Abstract.html`. 4

L. Caccia, R. Aljundi, N. Asadi, T. Tuytelaars, J. Pineau, and E. Belilovsky. New insights on reducing abrupt representation change in online continual learning. In *International Conference on Learning Representations*, 2022. 9

Arslan Chaudhry and others. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019. URL https://arxiv.org/abs/1902.10486. 3, 4

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *International Conference on Learning Representations*, 2019a. 1, 4

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K. Dokania, Philip H. S. Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019b. 1

Andrea Cossu, Gabriele Graffieti, Lorenzo Pellegrini, Davide Maltoni, Davide Bacciu, Antonio Carta, and Vincenzo Lomonaco. Is class-incremental enough for continual learning? *Frontiers in Artificial Intelligence*, 5:829842, 2022. doi: 10.3389/frai.2022.829842. URL https://www.frontiersin.org/articles/10.3389/frai.2022.829842. 3, 4

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3366–3385, 2022. doi: 10.1109/TPAMI.2021.3057446. 3, 4

Natalia Díaz-Rodríguez, Vincenzo Lomonaco, David Filliat, and Davide Maltoni. Don't forget, there is more than forgetting: new metrics for continual learning. *arXiv preprint arXiv:1810.13166*, 2018. URL https://arxiv.org/abs/1810.13166. 4

Rui Gao and Weiwei Liu. Red alarm: Controllable backdoor attack in continual learning. SSRN preprint, 2024. URL https://ssrn.com/abstract=5004179. 4

Rui Gao and Weiwei Liu. Red alarm: Controllable backdoor attack in continual learning. *Neural Networks*, 2025. URL https://www.sciencedirect.com/science/article/pii/S0893608025003582. Online first. 4

Zhen Guo, Abhinav Kumar, and Reza Tourani. Persistent backdoor attacks in continual learning. In *Proc. of the USENIX Security Symposium*, 2025a. 2

Zhen Guo, Abhinav Kumar, and Reza Tourani. Persistent backdoor attacks in continual learning. In *Proceedings of the 34th USENIX Security Symposium (USENIX Security 2025)*, 2025b. URL https://www.usenix.org/system/files/conference/usenixsecurity25/sec25cycle1-prepub-670-guo-zhen.pdf. 4

Gyojin Han, Jaehyun Choi, Hyeong Gwon Hong, and Junmo Kim. Data poisoning attack aiming the vulnerability of continual learning. In *IEEE International Conference on Image Processing (ICIP)*, pp. 1905–1909, 2023. 2

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, June 2016. 9

Jingrui Hou, Georgina Cosma, and Axel Finke. Advancing continual lifelong learning in neural information retrieval: definition, dataset, framework, and empirical evaluation. *arXiv preprint arXiv:2308.08378*, 2024. URL https://arxiv.org/abs/2308.08378. Preprint of the version later appearing in *Information Sciences*. 4

Wei-Cheng Huang, Chun-Fu Chen, and Hsiang Hsu. Ovor: Oneprompt with virtual outlier regularization for rehearsal-free class-incremental learning, 2024. URL https://arxiv.org/abs/2402.04129. 6

Ferenc Huszár. Note on the quadratic penalties in elastic weight consolidation. *Proceedings of the National Academy of Sciences*, 115(11):E2496–E2497, 2018. doi: 10.1073/pnas.1717042115. URL https://www.pnas.org/doi/10.1073/pnas.1717042115. 4

Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE Symposium on Security and Privacy (SP)*, pp. 19–35. IEEE, 2018. doi: 10.1109/SP.2018.00057. URL https://arxiv.org/abs/1804.00308. 2

Svante Janson and Svante Linusson. The probability of the alabama paradox. *Journal of Applied Probability*, 49(3):773–794, 2012. doi: 10.1239/jap/1346955333. URL https://doi.org/10.1239/jap/1346955333. 7

Siteng Kang, Zhan Shi, and Xinhua Zhang. Poisoning generative replay in continual learning to promote forgetting. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, pp. 15769–15785, 2023. 2

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, and *et al.* Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. 1, 3, 4

Kummari Naveen Kumar, Chalavadi Krishna Mohan, and Linga Reddy Cenkeramaddi. The impact of adversarial attacks on federated learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(5):2672–2691, 2024. doi: 10.1109/TPAMI.2023.3322785. 5

Lilly Kumari, Shengjie Wang, Tianyi Zhou, and Jeff A. Bilmes. Retrospective adversarial replay for continual learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 28530–28544, 2022. URL https://papers.nips.cc/paper_files/paper/2022/file/b6ffbbacbe2e56f2ec9a0da907382b4a-Paper-Conference.pdf. 4

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2021. doi: 10.1109/TPAMI.2021.3057446. 1

Huayu Li and Gregory Ditzler. Targeted data poisoning attacks against continual learning neural networks. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2022. doi: 10.1109/IJCNN55064.2022.9892774. 4, 13

Vincenzo Lomonaco and Davide Maltoni. Core50: a new dataset and benchmark for continuous object recognition. In Sergey Levine, Vincent Vanhoucke, and Ken Goldberg (eds.), *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pp. 17–26. PMLR, 13–15 Nov 2017. URL https://proceedings.mlr.press/v78/lomonaco17a.html. 9

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, pp. 6467–6476, 2017. 1, 4

Zheda Mai and contributors. online-continual-learning (code repository). GitHub, 2020. URL https://github.com/RaptorMai/online-continual-learning. Code accompanying Online Class-Incremental CL with Adversarial Shapley Value (AAAI 2021). 4

Zheda Mai, Ruiwen Li, Hyunwoo Kim, and Scott Sanner. Supervised contrastive replay: Revisiting the nearest class mean classifier in online class-incremental continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 3589–3599, June 2021. 9

Hongseok Namkoong and John C. Duchi. Stochastic gradient methods for distributionally robust optimization with f-divergences. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 2208–2216, 2016. URL https://proceedings.neurips.cc/paper/2016/file/4588e674d3f0faf985047d4c3f13ed0d-Paper.pdf. 2

Ameya Prabhu, Philip H. S. Torr, and Puneet K. Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *Computer Vision – ECCV 2020*, volume 12347 of *Lecture Notes in Computer Science*, pp. 524–540. Springer, 2020. doi: 10.1007/978-3-030-58536-5_31. URL https://link.springer.com/chapter/10.1007/978-3-030-58536-5_31. 4

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/fa7cdfad1a5aaf8370ebeda47a1ff1c3-Paper.pdf. 9

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *Proceedings of the International Conference on Learning Representations (ICLR) 2016*, 2016. URL https://arxiv.org/abs/1511.05952. Poster. 1, 2

Aspen D. Shen, Sanjoy Dasgupta, and Saket Navlakha. Reducing catastrophic forgetting with associative learning. *Neural Computation*, 35(11):1797–1817, 2023. URL https://direct.mit.edu/neco/article/35/11/1797/117579/. See also author PDF. 4

Aman Sinha, Hongseok Namkoong, and John C. Duchi. Certifying some distributional robustness with principled adversarial training. In *International Conference on Learning Representations (ICLR)*, 2018. doi: 10.48550/arXiv.1710.10571. URL https://openreview.net/forum?id=Hk6kPgZA-. 2

James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, Paola Cascante-Bonilla, Donghyun Kim, Ali Rostami, and Zsolt Kira. Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11909–11919, 2023. 6

Stanford. Tiny imagenet challenge (cs231n). https://www.kaggle.com/c/tiny-imagenet, 2015. Accessed: 2025-10-11. 9

Jacob Steinhardt, Pang Wei Koh, and Percy Liang. Certified defenses for data poisoning attacks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 3517–3529, 2017. doi: 10.48550/arXiv.1706.03691. URL https://proceedings.neurips.cc/paper/2017/hash/9d7311ba459f9e45ed746755a32dcd11-Abstract.html. 2

Muhammad Umer, Ajmal Mian, et al. Pacol: Poisoning attacks against continual learners. *arXiv preprint arXiv:2311.10919*, 2023. 2, 13

Gido M. van de Ven, Tinne Tuytelaars, and Andreas S. Tolias. Three types of incremental learning. *Nature Machine Intelligence*, 4:1185–1197, 2022. doi: 10.1038/s42256-022-00568-3. URL https://www.nature.com/articles/s42256-022-00568-3. 3, 4

Gido M. van de Ven, Tinne Tuytelaars, and Andreas S. Tolias. Three types of incremental learning. *Nature Machine Intelligence*, 4:1185–1197, 2022. doi: 10.1038/s42256-022-00568-3. URL https://doi.org/10.1038/s42256-022-00568-3. 1

Qian Wang et al. Continual adversarial defense. *arXiv preprint arXiv:2312.09481*, 2023. URL https://arxiv.org/abs/2312.09481. 4

Fei Ye and Adrian G. Bors. Online task-free continual learning via dynamic expandable memory distribution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11254–11264, June 2025. doi: 10.1109/CVPR57264.2025.11254. URL https://openaccess.thecvf.com/content/CVPR2025/papers/Ye_Online_Task-Free_Continual_Learning_via_Dynamic_Expandable_Memory_Distribution_CVPR_2025_paper.pdf. 1

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 3987–3995. JMLR.org, 2017. 9