RELIABLE DEPLOYMENT-TIME LEARNING: FROM LABEL ACQUISITION TO ERROR RECOVERY

Anonymous authors
Paper under double-blind review

ABSTRACT

This paper studies the reliability of deployment-time learning. As AI systems increasingly need to acquire new knowledge after deployment, a learning agent must detect novel objects or classes unseen during training, acquire their labels from human users or other external knowledge sources, and incrementally incorporate this new knowledge into the deployed model. However, externally acquired labels may not always be directly suitable for learning: they may use synonyms or hypernyms of the intended class names, contain errors, or even be maliciously provided. Moreover, learning new classes should not compromise previously acquired knowledge, and errors that have already been learned should be recoverable. We address these challenges based on the learning method of *incremental Linear Discriminant Analysis* (iLDA) over representations produced by strong pretrained foundation models. iLDA is particularly suitable for deployment-time learning because it supports efficient online incremental updates while preserving previously acquired knowledge with high accuracy. Building on this foundation, we develop methods for reliable label acquisition that resolve semantically related labels and detect potentially incorrect labels before learning, as well as an efficient recovery mechanism for removing the influence of incorrect labels that have already been incorporated into the model. Experimental results on multiple benchmark datasets demonstrate the effectiveness of the proposed methods across label acquisition, incremental learning, and error recovery, providing a practical framework for reliable learning after deployment.

1 INTRODUCTION

Machine learning models are traditionally trained offline using a carefully curated training dataset and then deployed with their parameters largely fixed. This paradigm assumes that the knowledge required after deployment is sufficiently represented in the training data. In many real-world applications, this assumption does not hold. The deployment environment may evolve and new objects or classes may emerge. Increasingly, we want the deployed model to be able to learn continuously in its application environment. Consider a deployed classification agent operating in a dynamic environment (Bendale & Boult, 2015; Kaymak et al., 2025). When a new object or class appears, the agent should first recognize that the object does not belong to any class it currently knows, i.e., detecting it as out-of-distribution (OOD). It can then query a human user, domain expert, or another knowledgeable external source for the ground-truth label and incrementally learn the newly labeled data. Repeating this process enables the system to expand its knowledge continually while operating in its deployment environment. We refer to this setting as *deployment-time learning*.

However, allowing an AI agent to modify its own knowledge after deployment introduces some important reliability challenges.

(1). Reliability of New Labels. When the agent queries a human user or an external source for ground-truth labels, the returned information may not always be directly suitable for learning: the provided labels may be synonyms or hypernyms of the intended class names, or they may contain errors. Blindly incorporating every provided label can therefore contaminate the model and adversely affect its future predictions. A reliable learning agent must be able to resolve such variations in class names and detect potentially incorrect labels before incorporating them into the model.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

(2). Catastrophic Forgetting (CF). As the agent incrementally learns new classes, the learning process should not degrade its ability to recognize previously learned classes. This is called *catastrophic forgetting* (McCloskey & Cohen, 1989). A reliable deployment-time learner must therefore acquire new knowledge while preserving the previously learned knowledge.

(3). Unlearning of Wrong Knowledge. No error detection mechanism can be expected to be perfect. Information initially believed to be correct may later be discovered to be wrong. In such cases, the system must be able to *unlearn* the incorrect knowledge. When the correct information becomes available, the system should further be able to relearn it efficiently.

In this work, we study these reliability challenges based on *incremental Linear Discriminant Analysis* (iLDA) over representations extracted from a strong pre-trained model (PTM) as iLDA has been shown to be one of the most effective approaches for continual learning and can achieve the accuracy of retraining or joint training using all previous and current data (Momeni et al., 2025) and is particularly suitable for deployment-time learning (Kaymak et al., 2025). With a frozen PTM, iLDA can update its class statistics incrementally without retraining the model from scratch. More importantly, it avoids CF caused by changes to the learned representation (Momeni et al., 2025).

We therefore focus our work on the other two challenges: (1) resolving class-name variations and detecting potentially incorrect labels, and (2) unlearning incorrect knowledge that has already been incorporated into the model. To address the first challenge, we leverage the statistical structure maintained by iLDA, a vision-language model, and agent–human interaction to resolve class-name variations and detect potential labeling errors. To address the second challenge, we exploit the sufficient statistics maintained by iLDA to efficiently remove the contributions of incorrectly labeled samples. Once the correct labels become available, these samples can be relearned under their corrected labels without retraining on all previously observed data. Importantly, this process enables the system to recover the same model that would have been obtained had the correct labels been provided from the beginning.

The main contributions of this work are summarized as follows:

- We formulate the reliability problem of deployment-time learning around three fundamental requirements (see above). To the best of our knowledge, this formulation has not been studied systematically in prior work.
- We propose a method for reliable label acquisition that tries to resolve class-name variations, and detects potentially incorrect labels before they are incorporated into the model.
- We develop an exact unlearning method that removes knowledge subsequently found to be incorrect and relearns it using corrected labels without retraining from scratch, recovering the same model that would have been obtained had the correct labels been provided from the beginning.
- We conduct extensive experiments on multiple benchmark datasets to evaluate the proposed framework and demonstrate its effectiveness. [1]

2 Related Work

Deployment-time continual learning. Traditional continual learning focuses on preventing CF (Parisi et al., 2019; Kirkpatrick et al., 2017; Rebuffi et al., 2017; Momeni et al., 2025), while out-of-distribution (OOD) detection identifies unseen categories (Hendrycks & Gimpel, 2017; Lee et al., 2018; Yang et al., 2024) without learning them. *Deployment-time learning* unifies both: an autonomous agent must detect novel inputs as OOD, query external sources for candidate labels, and expand its classifier incrementally without forgetting (Bendale & Boult, 2015; Kaymak et al., 2025). Our work is different as we deal with reliability issues in deployment-time learning.

Where the labels come from. Answers from several people can be combined by estimating how reliable each one is (Dawid & Skene, 1979; Whitehill et al., 2009; Welinder et al., 2010; Zheng et al., 2017; Snow et al., 2008), but those methods also assume one shared list of labels, whereas the users of a deployed agent type free text. A second opinion instead comes from a vision–language model such as CLIP (Radford et al., 2021; Ilharco et al., 2021), which scores an image against any

[1] The code is provided in the *Supplementary Materials*.

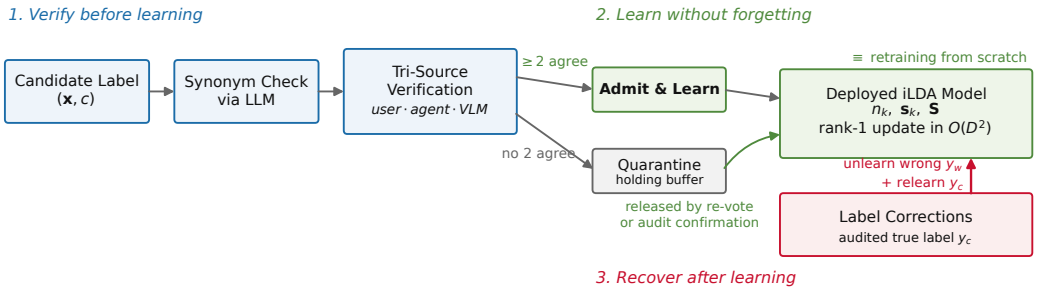


Figure 1: The three problems as one loop. A term for a class label returned by a user is judged against the agent's own prediction and a vision–language model, and a term the agent has never heard is first put to a language model as a possible synonym of one it already holds (Section 4). What survives is learned as counts and sums (Section 3), and what has been learned stays revisable, because the same statistics run backwards (Section 5).

written name; we show that this works when the checker is given the correct name of every class in advance, and not when it may use only the words the agent has actually heard.

Machine unlearning. Removing the influence of training data has been studied mostly for privacy: exact methods retrain on shards (Bourtoule et al., 2021) or exploit the structure of the model (Cao & Yang, 2015; Ginart et al., 2019), and approximate ones bound the remaining influence instead of removing it (Guo et al., 2020); see Nguyen et al. (2025). We use unlearning not to satisfy a deletion request but to keep a deployed model consistent with what its operator now believes the labels to be, which the stored sums make exact at two rank-one updates per sample. Learning from human feedback raises reliability questions even when the feedback is honest (Amodei et al., 2016; Christiano et al., 2017; Ouyang et al., 2022; Casper et al., 2023), and poisoned data is a separate threat (Biggio et al., 2012; Steinhardt et al., 2017; Carlini et al., 2024); we model ordinary human error and variation in wording, with one consistently wrong user as the worst case.

3 Incremental LDA on a Frozen Pre-trained Model

Our agent learns at deployment time with iLDA on top of a frozen pre-trained model (PTM). As illustrated in Figure 1, our deployment pipeline unifies the three requirements into a continuous loop: detecting wrong knowledge before it is learned (Section 4), learning incrementally without catastrophic forgetting (CF) (see below), and exact post-hoc unlearning recovery (Section 5).

Features. An input $\mathbf{x}$ is mapped by a frozen PTM f_θ to $\boldsymbol{\phi} = f_\theta(\mathbf{x}) \in \mathbb{R}^d$. Because f_θ is never updated, what a feature means does not change while the agent is deployed. A fixed random matrix $\mathbf{W} \in \mathbb{R}^{d\times D}$ with entries drawn from $\mathcal{N}(0,1)$ and an element-wise ReLU $\sigma(\cdot)$ then expand the representation to $D > d$ dimensions

$$\mathbf{z} = \sigma(\boldsymbol{\phi}^\top \mathbf{W}) \in \mathbb{R}^D. \tag{1}$$

This is called *random projection*, which improves linear separability and, like the PTM, learns no parameter of its own (Huang et al., 2011; McDonnell et al., 2023). The projected feature is then L_2-normalised onto the unit hypersphere,

$$\tilde{\mathbf{z}} = \frac{\mathbf{z}}{\|\mathbf{z}\|_2} \in \mathbb{S}^{D-1}. \tag{2}$$

Normalisation matters at deployment time. A class that appears after deployment is represented by a handful of samples, and removing the magnitude variation the random projection introduces leaves class statistics that are stable at that sample size.

What the model stores. For each class k the learner keeps a count and a running sum of normalised features, from which the class direction follows:

$$n_k = |\{i : y_i = k\}|, \qquad \mathbf{s}_k = \sum_{i:y_i=k} \tilde{\mathbf{z}}_i, \qquad \boldsymbol{\mu}_k = \frac{\mathbf{s}_k}{\|\mathbf{s}_k\|_2}. \tag{3}$$

Here $\boldsymbol{\mu}_k$ is the angular centre of class k rather than the arithmetic mean, the maximum-likelihood direction of a von Mises–Fisher distribution (Mardia & Jupp, 1999), which again is what makes the estimate usable from few samples.

Based on LDA, all classes share one covariance. Writing the *centroid correction* of class k as

$$\mathbf{C}_k = \big(n_k - 2\|\mathbf{s}_k\|_2\big)\,\boldsymbol{\mu}_k\boldsymbol{\mu}_k^\top, \tag{4}$$

expanding the within-class scatter and substituting $\mathbf{s}_k = \|\mathbf{s}_k\|_2\boldsymbol{\mu}_k$ gives

$$\mathbf{S} = \sum_i \big(\tilde{\mathbf{z}}_i - \boldsymbol{\mu}_{y_i}\big)\big(\tilde{\mathbf{z}}_i - \boldsymbol{\mu}_{y_i}\big)^\top = \sum_i \tilde{\mathbf{z}}_i\tilde{\mathbf{z}}_i^\top + \sum_k \mathbf{C}_k, \qquad \boldsymbol{\Sigma} = \frac{1}{N}\mathbf{S}, \tag{5}$$

with $N = \sum_k n_k$. Equation 5 is the usual within-class scatter in a form that needs no data: a sum of per-sample outer products plus one correction per class, each computable from the count and sum that class already keeps. This decomposition is what makes both of the updates below possible.

Learning one sample. To add a labelled sample $(\tilde{\mathbf{z}}, k)$, the learner first evaluates $\mathbf{C}_k^{\text{old}}$ from Equation 4 at the current statistics, updates

$$n_k \leftarrow n_k + 1, \qquad \mathbf{s}_k \leftarrow \mathbf{s}_k + \tilde{\mathbf{z}}, \qquad N \leftarrow N + 1, \tag{6}$$

re-evaluates $\mathbf{C}_k^{\text{new}}$ at the new ones, and sets

$$\mathbf{S} \leftarrow \mathbf{S} + \tilde{\mathbf{z}}\tilde{\mathbf{z}}^\top - \mathbf{C}_k^{\text{old}} + \mathbf{C}_k^{\text{new}}. \tag{7}$$

The pair $-\mathbf{C}_k^{\text{old}} + \mathbf{C}_k^{\text{new}}$ swaps the stale correction for the fresh one, so the resulting $\boldsymbol{\Sigma}$ is **identical** to recomputing Equation 5 from scratch over all N samples. Only the class that the sample joins is touched, the cost is $O(D^2)$, no data is kept beyond the statistics themselves, and a class seen for the first time simply starts a new pair $(n_k, \mathbf{s}_k)$.

Inference. The shared covariance is regularised towards the identity, $\boldsymbol{\Lambda} = ((1-\epsilon)\boldsymbol{\Sigma} + \epsilon\mathbf{I})^{-1}$ (see Appendix A), and a sample is scored against each class by the usual discriminant

$$h_k(\mathbf{z}) = \mathbf{z}^\top\boldsymbol{\Lambda}\boldsymbol{\mu}_k - \tfrac{1}{2}\,\boldsymbol{\mu}_k^\top\boldsymbol{\Lambda}\boldsymbol{\mu}_k. \tag{8}$$

The same score detects what the model does not yet know: a sample whose best score falls below a *threshold* is taken to be out-of-distribution (OOD), which is how iLDA over a frozen PTM is used for open-world recognition (Kaymak et al., 2025). We use it in exactly that role — it is what decides which samples the agent asks about (Section 4) — and, being a function of the same statistics, it moves with the model at no extra cost.

No forgetting. Since each class owns its count and sum, and Equations 6–7 touch one class at a time and incrementally updated $\boldsymbol{\Sigma}$ is identical to computing it from scratch over all samples, there is thus no forgetting. iLDA of this kind has been shown to match joint training over a frozen PTM in continual learning (Momeni et al., 2025) and to support learning after deployment (Kaymak et al., 2025). Our additions of random projection and normalization do not change it. We take forgetting as settled and do not study it further; Appendix B reports the measurement for our own runs.

4 Name Variation Resolution and Wrong Label Detection

Deployment-Time Label Modalities and Fragmentation. When the deployed agent flags an observation $\mathbf{x}$ as novel, it queries an external source (e.g., a human user) and receives a candidate label c. In real-world environments, external inputs are not guaranteed to follow a clean, canonical taxonomy. We categorize received labels into four distinct modalities:

1. **Canonical Label**: The correct, standard class name for the observed object.
2. **Label Error**: An incorrect class name (e.g., "apple" for a tomato) resulting from human error, lack of expertise, or adversarial intent.
3. **Synonym / Aliasing**: A valid alternative name for an existing class (e.g., "aubergine" for an eggplant).
4. **Hypernym / Foreign Term**: A broader categorical term (e.g., "vegetable") or an uninterpretable foreign token (e.g., "lex0042").

Crucially, while case (2) represents false information, cases (3) and (4) carry valid knowledge. However, an autonomous learner that indiscriminately creates a new class for every novel string splits single visual concepts across multiple categories. We term this pathology *label-space fragmentation*. Fragmentation severely impairs deployment utility: test samples belonging to a single underlying class are divided among fragmented groups, allowing only one group to match the ground truth and leading to steep drops in classification accuracy. An effective deployment-time safeguard must therefore separate genuine label errors from lexical variations.

Algorithm 1 Term-First Label Verification for feature $\tilde{\mathbf{z}}$ of Sample x with User Term c

1 **if** c is an unfamiliar term **then**
  Ask LLM if c is synonymous with any of the 8 nearest known class names;
  if synonym match found with class k, map $c \leftarrow k$
2 $y \leftarrow \arg\max_k h_k(\mathbf{z})$; $A(w) \leftarrow$ VLM-Accept$(\mathbf{x}, w)$
3 **if** $A(c)$ **then** LEARN$(\tilde{\mathbf{z}}, c)$ *(Consensus: User + VLM)*
4 **else if** $c \in$ KnownClasses **and** $y = c$ **then** LEARN$(\tilde{\mathbf{z}}, c)$ *(Consensus: User + Agent)*
5 **else if** $A(y)$ **then** LEARN$(\tilde{\mathbf{z}}, y)$ *(Consensus: Agent + VLM)*
6 **else if** $c \notin$ KnownClasses **then**
  Query user: "Is this y, rather than c?"; **if** confirmed, LEARN$(\tilde{\mathbf{z}}, y)$ **else** QUARANTINE(x)
7 **else** QUARANTINE(x) *(Disagreement: Hold in quarantine buffer)*
8 **Audit Phase (Optional):** When verification budget permits, audit quarantined samples.

4.1 TERM-FIRST VERIFICATION STRATEGY

Our primary detection framework combines cross-modal consensus with semantic language reasoning to verify candidate labels before updating model statistics.

Tri-Source Consensus. For an OOD sample $\mathbf{x}$ that needs to be labeled, the agent collects three distinct assessments: 1. *External Source*: The candidate string c provided by the user. 2. *Agent Self-Prediction*: The class $y = \arg\max_k h_k(\mathbf{z})$ from Equation 8 (evaluated without OOD thresholding to extract the model's closest visual match). 3. *Zero-Shot VLM Verification*: A binary verification decision $A(c)$ from a vision-language model (CLIP), which checks whether the image-text cosine similarity between x and prompt "a photo of a c" exceeds a validation threshold calibrated prior to deployment. Querying binary acceptance on single terms prevents the VLM from hallucinating spurious classes when unconstrained.

LLM Synonym Disambiguation. When an unfamiliar candidate term c is observed for the first time, the agent queries a large language model (GPT-5.6-Terra) to determine if c is a synonym of any known class name. To minimize API latency, the query is restricted to the 8 nearest known class names in CLIP text embedding space. If a synonym match is confirmed, c is merged into the existing class representation, unifying the linguistic alias and directly preventing label-space fragmentation.

Verification and Quarantine Rule. Algorithm 1 formalizes the Term-First verification protocol. The sample is admitted only when at least two independent sources agree ($\geq 2/3$ majority vote).

4.2 OBJECT-FIRST VERIFICATION STRATEGY

As a complementary strategy, we examine an *Object-First* clustering detector that groups incoming observations based on visual feature consistency before assigning semantic class names. It computes Hotelling's two-sample T^2 statistic between the sample representation and candidate cluster centroids:

$$T^2(a, b) = \frac{n_a n_b}{n_a + n_b}(\boldsymbol{\mu}_a - \boldsymbol{\mu}_b)^\top \boldsymbol{\Lambda}(\boldsymbol{\mu}_a - \boldsymbol{\mu}_b). \tag{9}$$

By clustering images visually prior to lexical assignment, incorrect user labels cannot contaminate established feature distributions. However, this strategy requires sufficient samples per class to construct reliable covariance estimates and exhibits sensitivity in fine-grained regimes. Complete mathematical derivations of the Object-First approach and its ablation experiments are provided in Appendix C and Appendix D, respectively.

5 EXACT UNLEARNING AND MODEL RECOVERY

Even with pre-learning verification, erroneous samples may bypass detection, or labels initially accepted in good faith may later be flagged during audits or crowd revocations. Furthermore, deployment errors in the open world span multiple granularities—from isolated mislabeled samples to structured taxonomy defects, such as late synonym discovery requiring class merges, source revocations, or batch audits. We address them with single-sample or batch *exact unlearning and recovery*.

Decoupling Feature Scatter from the Label Space. Recall from Equation 5 that the total within-class scatter decomposes as:

$$\mathbf{S} = \underbrace{\sum_{i=1}^{N} \tilde{\mathbf{z}}_i \tilde{\mathbf{z}}_i^\top}_{\mathbf{T}} + \sum_k \phi(n_k, \mathbf{s}_k), \qquad \phi(n, \mathbf{s}) = \Big(\frac{n}{\|\mathbf{s}\|_2^2} - \frac{2}{\|\mathbf{s}\|_2} \Big) \mathbf{s}\mathbf{s}^\top, \tag{10}$$

where $\phi(n_k, \mathbf{s}_k) = \mathbf{C}_k$ is the class centroid correction (Eq. 4), with $\phi(0, \mathbf{0}) = \mathbf{0}$. This formulation exposes a fundamental architectural decoupling: the uncentered sample second moment $\mathbf{T}$ depends strictly on *which* feature vectors were observed, completely independent of their assigned class labels. The label assignments enter exclusively through the per-class summary statistics $(n_k, \mathbf{s}_k)$, each through a single rank-one term $\phi(n_k, \mathbf{s}_k)$. Because both $\mathbf{T}$ and $\sum_k \phi$ are associative sums, the model parameters depend solely on the final multiset of retained labeled samples, guaranteeing *order independence*.

Proposition 1 (Exact Unlearning and Edit Rule)**.** *Let a model modification add a sample batch $\mathbf{Z}_+$ and withdraw $\mathbf{Z}_-$ (stacked row-wise, either of which may be empty, a single sample, or a batch), while updating the summary statistics of an affected class set $\mathcal{A}$ from $(n_k, \mathbf{s}_k)$ to $(n'_k, \mathbf{s}'_k)$. The updated statistics and scatter matrix are given exactly by:*

$$N' = N + |\mathbf{Z}_+| - |\mathbf{Z}_-|, \qquad \mathbf{S}' = \mathbf{S} + \mathbf{Z}_+^\top \mathbf{Z}_+ - \mathbf{Z}_-^\top \mathbf{Z}_- + \sum_{k \in \mathcal{A}} \big[\phi(n'_k, \mathbf{s}'_k) - \phi(n_k, \mathbf{s}_k) \big], \tag{11}$$

with updated covariance $\mathbf{\Sigma}' = \frac{1}{N'} \mathbf{S}'$. The resulting parameters $(N', \mathbf{S}', \{n'_k, \mathbf{s}'_k\})$ are algebraically identical to retraining from scratch on the edited multiset.

The proof is immediate from Eq. 10: $\mathbf{T}$ is adjusted only by samples entering or exiting the model, while ϕ is invariant for all unaffected classes $k \notin \mathcal{A}$.

Single-Sample Downdating and Relearning. To unlearn an isolated erroneous sample $(\tilde{\mathbf{z}}_j, y_w)$, Proposition 1 instantiates with $\mathbf{Z}_+ = \emptyset$, $\mathbf{Z}_- = \{\tilde{\mathbf{z}}_j\}$, and $\mathcal{A} = \{y_w\}$. The statistics of class y_w are downdated by subtraction:

$$n^*_{y_w} = n_{y_w} - 1, \qquad \mathbf{s}^*_{y_w} = \mathbf{s}_{y_w} - \tilde{\mathbf{z}}_j, \qquad N^* = N - 1. \tag{12}$$

(If $n^*_{y_w} = 0$, class y_w is deleted from the model.) The scatter matrix update instantiates as:

$$\mathbf{S} \leftarrow \mathbf{S} - \tilde{\mathbf{z}}_j \tilde{\mathbf{z}}_j^\top - \phi(n_{y_w}, \mathbf{s}_{y_w}) + \phi(n^*_{y_w}, \mathbf{s}^*_{y_w}), \qquad \mathbf{\Sigma} \leftarrow \frac{1}{N^*} \mathbf{S}, \tag{13}$$

exhibiting exact structural symmetry with forward learning (Eq. 7) in $O(D^2)$ time. When relearning sample j to true class y_c, the sample remains in the model, so outer products cancel ($-\tilde{\mathbf{z}}_j \tilde{\mathbf{z}}_j^\top + \tilde{\mathbf{z}}_j \tilde{\mathbf{z}}_j^\top = \mathbf{0}$); relearning thus updates four rank-one corrections without modifying $\mathbf{T}$.

Batch Removal and Relearning. While sample downdating handles isolated removals, deployment errors are often structured: (i) *Source revocation*: an untrusted user is revoked, requiring all their data to be withdrawn or re-assigned; (ii) *Late synonym discovery*: two established classes describe the same entity (e.g., "aubergine" and "eggplant"), requiring a class merge; (iii) *Refuted links*: two distinct classes erroneously joined by an LLM must be split; and (iv) *Batch audit*: human reviewers return corrected labels for a batch of samples.

Given a batch $\mathbf{Z}_-$ of m erroneous samples in class k, Proposition 1 subtracts the feature block $\mathbf{Z}_-^\top \mathbf{Z}_-$ directly and replaces $\phi(n_k, \mathbf{s}_k)$ with $\phi(n_k - m, \mathbf{s}_k - \sum_{\mathbf{z} \in \mathbf{Z}_-} \mathbf{z})$ in a single step, bypassing $O(mD^2)$ sequential loops. For batch relearning (m samples with sum $\mathbf{b}$ moved from class a to c), data scatter is strictly invariant ($\mathbf{Z}_+ = \mathbf{Z}_- = \mathbf{0}$). The update modifies only class summaries $(n_a, \mathbf{s}_a) \mapsto (n_a - m, \mathbf{s}_a - \mathbf{b})$ and $(n_c, \mathbf{s}_c) \mapsto (n_c + m, \mathbf{s}_c + \mathbf{b})$ in $O(mD + D^2)$ time, completely avoiding feature Gram matrix products. The same relabel also lets the agent correct itself during deployment without querying anyone: every R steps it re-runs the vote of Algorithm 1 on decisions that rested on its own earlier prediction and relabels the samples whose vote has changed (see Appendix F).

Data-Free Class Merging. For merging class a into c (case ii), $\mathbf{Z}_+ = \mathbf{Z}_- = \mathbf{0}$ and the update simplifies to:

$$\mathbf{S}' = \mathbf{S} - \phi(n_a, \mathbf{s}_a) - \phi(n_c, \mathbf{s}_c) + \phi(n_a + n_c, \mathbf{s}_a + \mathbf{s}_c). \tag{14}$$

Remarkably, class merging requires *no sample representations at all*—it operates purely on stored class summary vectors in $O(D^2)$ time. This allows privacy-critical or memory-constrained agents that discard raw features to resolve taxonomy fragmentation post-hoc. Similarly, splitting a refuted link requires only the sum vector of the migrating subset.

6 EXPERIMENTS

Datasets and Streaming Setup. We conduct experiments on four diverse benchmarks: 1. **CIFAR-100** (Krizhevsky, 2009): 100 classes, partitioned into 50 base classes and 50 novel/OOD classes. 2. **TinyImageNet** (Le & Yang, 2015): 200 classes of images, split into 100 base and 100 novel classes. 3. **ImageNet-R** (Hendrycks et al., 2021): 200 classes containing artistic renditions, paintings, and cartoons, split into 100 base and 100 novel classes. 4. **CUB-200-2011** (Wah et al., 2011): 200 fine-grained bird species, split into 100 base and 100 novel classes. For all datasets, 90% of base class data are used for pre-deployment training, while the remaining 10% of base class data and 10% of novel class data form the post-deployment streaming sequence. Since OOD detection is not the focus of this paper, we assume perfect OOD detection throughout our study.

Simulated Multi-User Crowd. In the main experiments, to model realistic human-in-the-loop interactions during deployment, candidate labels are provided by a simulated crowd of 11 diverse users (detailed in Table 6, Appendix E): - 4 *Reliable* users (using canonical class names); - 2 *Sloppy* users (confusing visually similar classes); - 2 *Synonym* users (providing valid synonyms such as "aubergine" for eggplant); - 1 *Hypernym* user (providing broader terms such as "solanaceous vegetable"); - 1 *Foreign* user (providing valid but out-of-vocabulary tokens such as "lex0042"); - 1 *Adversarial* user (systematically providing deliberately inverted labels). This mixture yields 21.2% erroneous labels, 16.5% synonyms, and 7.7% hypernyms on CIFAR-100. Separately, to isolate vocabulary variation from label errors under controlled conditions, we evaluate a *vocabulary ladder* (Figure 2): each rung uses a dedicated 12-user crowd (2 sloppy and 1 adversarial users fixing the baseline label error rate at $\sim$20%, and 9 reliable users), where 0, 2, 4, 6, or 8 of the 9 reliable users progressively adopt alternative synonyms while keeping underlying accuracy unchanged.

Evaluation Metrics. 1. **Matched Accuracy (%)**: Following the open-world protocol, each predicted class group is assigned to a ground-truth class via Hungarian matching (Kuhn, 1955). When a true class is fragmented across multiple groups, only one group can match, penalizing fragmentation. 2. **Label-Space Inflation**: The ratio of learned class groups to ground-truth classes. An inflation of 1.00 indicates perfect taxonomy recovery; values above 1.00 reflect fragmentation, while values below 1.00 indicate class merging. 3. **Relative Covariance Gap**: $\|\mathbf{\Sigma}_{\text{recovered}} - \mathbf{\Sigma}_{\text{oracle}}\|_F / \|\mathbf{\Sigma}_{\text{oracle}}\|_F$, quantifying the numerical exactness of unlearning relative to retraining from scratch.

Baselines and Models. We compare against: *No Curation* (unconditionally learning all received labels), *Redundant Majority Voting* (querying two additional annotators), *VLM Veto* (letting CLIP overrule user labels among observed terms), and the *Agreement Gate* (tri-source consensus without LLM synonym resolution). As reference bounds, we report *VLM Veto (All Terms)* (given the true class names in advance), *Clean Oracle* (trained incrementally on ground-truth labels), and *Joint Fine-Tuning (JFT)* (full offline backbone fine-tuning). All methods use a frozen DINOv2-ViT-B/14 backbone ($d = 768$) (Oquab et al., 2024) with random projection dimension $D = 5000$.

6.1 NAME RESOLUTION AND WRONG LABEL DETECTION

Table 1 compares all methods across the four benchmark datasets under the 11-user crowd.

Reference Bounds: Clean Oracle and Joint Fine-Tuning. The *Clean Oracle* defines the incremental performance ceiling when all streaming data arrive with canonical ground-truth labels. iLDA on frozen DINOv2 representations closely tracks offline *Joint Fine-Tuning (JFT)* on CIFAR-100, TinyImageNet, and ImageNet-R, confirming that closed-form recursive estimation avoids catastrophic forgetting without iterative retraining. Crucially, on CUB-200, iLDA outperforms joint fine-tuning, showing its robustness under class imbalance and low sample support. When provided with ground-truth class names in advance, *VLM Veto* also approaches the oracle ceiling, though such oracle vocabulary access is unavailable in autonomous open-world deployment. The exception is CUB-200, where CLIP is weak on fine-grained birds: a species that CLIP never picks for any of its few stream images gets no class, so its images are absorbed by look-alike species (inflation below 1.00).

Method	Matched accuracy (%)				Label-space inflation			
	C100	TinyIN	IN-R	CUB	C100	TinyIN	IN-R	CUB
No curation	58.8 ±1.6	57.2 ±1.1	58.3 ±0.5	73.6 ±1.5	3.26	3.25	2.69	1.42
Redundant majority	59.0 ±1.6	57.4 ±1.0	58.5 ±0.6	74.0 ±1.6	3.26	3.25	2.67	1.41
VLM veto	75.1 ±0.9	73.2 ±0.6	67.9 ±0.7	70.7 ±0.6	2.07	2.15	1.76	0.95
Agreement gate	70.7 ±0.7	68.2 ±0.9	66.5 ±0.9	77.0 ±0.4	2.47	2.44	1.87	1.07
Term-first check (ours)	76.1 ±1.3	73.3 ±1.0	**68.9 ±0.3**	**77.3 ±0.4**	2.08	2.01	**1.57**	**1.04**
Object-first check (ours)	**79.2 ±1.2**	**79.4 ±0.3**	65.7 ±0.7	71.2 ±1.2	**1.16**	**1.16**	1.78	1.41
VLM veto, given all terms	85.1 ±0.0	82.8 ±0.2	75.6 ±0.3	71.4 ±0.5	1.00	1.00	1.00	0.92
Clean oracle	88.1 ±0.0	84.9 ±0.2	78.0 ±0.4	83.7 ±0.4	1.00	1.00	1.00	1.00
Joint fine-tuning	89.7 ±0.1	85.9 ±0.1	78.2 ±0.2	80.9 ±0.1	—	—	—	—

Table 1: Matched accuracy (mean ±sd over 3 seeds) and label-space inflation, where 1.00 means the agent holds exactly one group per class. The last three rows are reference points a deployed agent cannot reach: one is given the term of every class, one every label, and the last is a backbone fine-tuned jointly on all classes with clean labels and no stream at all, which has no label space to inflate. Bold marks the best deployable method per column.

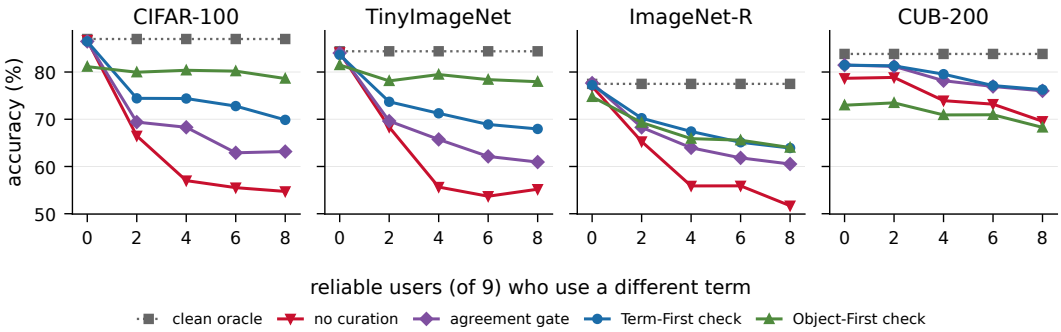


Figure 2: The vocabulary ladder analysis across benchmark datasets. Moving horizontally across rungs, reliable users progressively adopt synonyms while maintaining identical underlying accuracy (∼20% error rate). The sharp accuracy drops illustrate the pure cost of vocabulary fragmentation.

Failure of Baselines Without Synonym Resolution. *No Curation* causes catastrophic performance collapse across all benchmarks, accompanied by severe label-space inflation. As demonstrated by the vocabulary ladder (Figure 2), this degradation is driven primarily by vocabulary variation rather than labeling errors: unmerged synonyms shatter true classes into redundant duplicate categories. Standard filtering strategies such as *Redundant Majority Voting* cannot prevent this fragmentation because multiple annotators often agree on a valid alternative term. Similarly, when *VLM Veto* is restricted to deployment-discovered terms, its performance deteriorates sharply; zero-shot vision-language queries evaluate candidate names in isolation, validating plausible synonyms as distinct classes. The *Agreement Gate* reliably filters out noisy labels, but confirmed synonyms are still admitted under separate strings, leaving the label space substantially fragmented.

Term-First vs. Object-First Verification. Both proposed strategies effectively arrest label-space fragmentation. In the **Term-First check**, integrating LLM synonym disambiguation unifies linguistic aliases before model updates, yielding marked accuracy gains over the Agreement Gate and substantially dampening label inflation. Term-First delivers the highest overall robustness across diverse visual domains, achieving the best performance among deployable methods on ImageNet-R and CUB-200. The **Object-First check** clusters visual features prior to assigning semantic labels. In data-dense regimes with abundant samples per class (CIFAR-100 and TinyImageNet), it achieves the highest accuracy and the tightest label space by bypassing linguistic variation entirely. This includes hypernyms: because a sample joins a group by appearance, a broader term such as "solanaceous vegetable" becomes one more, outvoted name candidate on the eggplant group rather than a class of its own, and a parent term used for two visually distinct groups is recorded as a hierarchy edge without merging them. However, when sample support per class is sparse, appearance-based clustering over-splits classes into disjoint visual clusters. Consequently, Term-First represents the superior general-purpose safeguard for unconstrained deployment.

Agent	After deployment → after repair (%)				Gap to oracle
	C100	TinyIN	IN-R	CUB	
No curation	58.8 → 88.1	57.2 → 84.9	58.3 → 78.0	73.6 → 83.7	+0.0
Term-First check (ours)	76.1 → 88.1	73.3 → 84.9	68.9 → 78.0	77.3 → 83.7	+0.0
+ re-vote, exact moves	76.7 → 88.1	73.7 → 84.9	69.2 → 78.0	77.8 → 83.7	+0.0
Learn now, exact moves	77.0 → 88.1	73.8 → 84.9	67.3 → 78.0	77.6 → 83.7	+0.0
Learn now, add copies	74.8 → 83.4	71.3 → 79.3	66.2 → 75.7	77.1 → 82.4	−3.5
Learn now, ignore corrections	73.7	71.2	65.5	77.1	—
Clean oracle	88.1	84.9	78.0	83.7	—

Table 2: Matched accuracy (%) at the end of deployment, with agents revising their own decisions and no audited labels, and after each agent is given the correct label of every sample and repaired (Section 5). The last column is the mean gap to the clean oracle after repair, in points, over the four datasets. An agent that ignores corrections keeps no consistent record of what it holds and cannot be repaired, so only its accuracy after deployment is shown. Agents are described in Appendix F.

6.2 Evaluation of Exact Unlearning and Model Recovery

Exact Convergence to the Clean Oracle. The last column gives each finished agent the correct label of every sample and repairs it with one relabel per (current, correct) class pair. Every agent that can unlearn then reaches the Clean Oracle exactly, and label-space inflation returns to exactly 1.00, as Proposition 1 predicts (in float64 to $\approx 10^{-15}$ relative covariance gap, Appendix B). *Learn now, add copies* stays 3.5 points short, because its stale copies cannot be removed.

Self-correction during deployment. Table 2 first reports accuracy at the end of deployment, with no audited labels. Besides the Term-First check, it includes agents that revise their own decisions every 100 steps with the batch relabel of Section 5 (Appendix F): + *re-vote* re-runs the vote on decisions that rested on the agent's own prediction, and the *learn-now* agents hold nothing back, learning every undecided sample at once under the user's term and correcting it later. Exact unlearning matters most when the agent does not hold data back. The learn-now agent spends 747 rank-one updates on corrections on CIFAR-100; applying them exactly keeps it level with the Term-First check, and beats adding corrected copies by $+1.6$ points ($[+1.0, +2.2]$) and $+2.0$ points ($[+1.4, +2.7]$) over ignoring revisions.

Anytime Servability and Computational Break-Even. As demonstrated by the recovery trajectories (Figure 3, Appendix G), accuracy converges monotonically toward the Clean Oracle as corrections are applied in random order. Because each rank-one downdate yields an algebraically exact LDA state, the classifier remains strictly valid and servable after every individual correction, eliminating maintenance downtime. A cost, runtime, and hardware analysis is provided in Appendix H.

7 Conclusion

In this work, we formulated the reliability of deployment-time learning around three core requirements: pre-learning label resolution and error detection, catastrophic forgetting prevention, and post-hoc unlearning recovery. We developed an integrated framework based on incremental Linear Discriminant Analysis over frozen foundation representations. We introduced a Term-First verification protocol combining cross-modal consensus and LLM synonym disambiguation to prevent label-space fragmentation, as well as an exact algebraic unlearning mechanism that eliminates erroneous knowledge in $O(D^2)$ time. Extensive empirical evaluations demonstrate that our framework effectively safeguards continuous learning systems against labeling errors and vocabulary variations, providing a practical foundation for reliable autonomous deployment.

Limitations. The users are simulated: their error rates, vocabularies and willingness to answer follow a fixed mixture, which lets us vary one factor at a time but is not a study with real people. The repair assumes corrected labels are available, and where they come from is outside the experiment, so what we measure is a capability rather than an automatic improvement. The synonym check relies on a commercial language model, though its answers are cached and shipped with the code. Finally, the exactness guarantees and absence of CF depend on maintaining a frozen PTM; extending exact unlearning to dynamically fine-tuned deep backbones represents an open challenge.

AI USE STATEMENT

Large language models are used in two ways in this work. First, one is part of the method itself: the term-first check asks a commercial language model (GPT-5.6-Terra) whether a newly heard term is a synonym of a known class name (Section 4.1), and CLIP serves as the vision–language model. All of the language model's answers are cached and released with the code, so the results can be reproduced without further queries. Second, the authors used an AI coding and writing assistant to help implement and debug the experiment code, draft and edit parts of the text, and prepare bibliography entries. The authors designed the study, reviewed and verified all code, derivations, reported numbers and citations, and take full responsibility for the content of the paper.

REFERENCES

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*, 2016.

Abhijit Bendale and Terrance Boult. Towards open world recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *International Conference on Machine Learning (ICML)*, 2012.

Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *IEEE Symposium on Security and Privacy (S&P)*, 2021.

Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *IEEE Symposium on Security and Privacy (S&P)*, 2015.

Nicholas Carlini, Matthew Jagielski, Christopher A Choquette-Choo, Daniel Paleka, Will Pearce, Hyrum Anderson, Andreas Terzis, Kurt Thomas, and Florian Tramèr. Poisoning web-scale training datasets is practical. In *IEEE Symposium on Security and Privacy (S&P)*, 2024.

Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, et al. Open problems and fundamental limitations of reinforcement learning from human feedback. *Transactions on Machine Learning Research*, 2023.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Alexander Philip Dawid and Allan M Skene. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Journal of the Royal Statistical Society: Series C*, 28(1):20–28, 1979.

Antonio Ginart, Melody Guan, Gregory Valiant, and James Y Zou. Making AI forget you: Data deletion in machine learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens van der Maaten. Certified data removal from machine learning models. In *International Conference on Machine Learning (ICML)*, 2020.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *IEEE International Conference on Computer Vision (ICCV)*, 2021.

Guang-Bin Huang, Hongming Zhou, Xiaojian Ding, and Rui Zhang. Extreme learning machine for regression and multiclass classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 42(2):513–529, 2011.

Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. OpenCLIP. In *Zenodo*, 2021.

Derda Kaymak, Gyuhak Kim, Tomoya Kaichi, Tatsuya Konishi, and Bing Liu. Learning after deployment. *Proceedings of the European Conference on Artificial Intelligence (ECAI-2025)*, 2025.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Harold W Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

Ya Le and Xuan Yang. Tiny ImageNet visual recognition challenge. Technical report, Stanford University, 2015.

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Kanti V Mardia and Peter E Jupp. *Directional Statistics*. John Wiley & Sons, 1999.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.

Mark D McDonnell, Dong Gong, Amin Parvaneh, Ehsan Abbasnejad, and Anton van den Hengel. RanPAC: Random projections and pre-trained models for continual learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Saleh Momeni, Sahisnu Mazumder, and Bing Liu. Continual learning using a kernel-based method over foundation models. In *AAAI-2025*, 2025.

Thanh Tam Nguyen, Thanh Trung Huynh, Phi Le Nguyen, Alan Wee-Chung Liew, Hongzhi Yin, and Quoc Viet Hung Nguyen. A survey of machine unlearning. *ACM Transactions on Intelligent Systems and Technology*, 2025.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, 2021.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. iCaRL: Incremental classifier and representation learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y Ng. Cheap and fast—but is it good? evaluating non-expert annotations for natural language tasks. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2008.

Jacob Steinhardt, Pang Wei Koh, and Percy Liang. Certified defenses for data poisoning attacks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The Caltech-UCSD birds-200-2011 dataset. Technical report, California Institute of Technology, 2011.

Peter Welinder, Steve Branson, Pietro Perona, and Serge Belongie. The multidimensional wisdom of crowds. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2010.

Jacob Whitehill, Ting-fan Wu, Jacob Bergsma, Javier Movellan, and Paul Ruvolo. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2009.

Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. Generalized out-of-distribution detection: A survey. *International Journal of Computer Vision*, 2024.

Yudian Zheng, Guoliang Li, Yuanbing Li, Caihua Shan, and Reynold Cheng. Truth inference in crowdsourcing: Is the problem solved? *Proceedings of the VLDB Endowment*, 10(5):541–552, 2017.

APPENDIX

A COVARIANCE REGULARIZATION AND SHRINKAGE TUNING

Table 3 summarizes the optimal covariance shrinkage parameter ϵ selected for each method family across the four benchmarks.

Method Family	CIFAR-100	TinyImageNet	ImageNet-R	CUB-200
Baselines and Term-First Check	10^{-5}	10^{-5}	10^{-5}	10^{-4}
Object-First Check	$3{\cdot}10^{-4}$	10^{-4}	10^{-4}	10^{-5}

Table 3: Covariance shrinkage parameter ϵ selected on validation Seed 0 from the candidate set $\{10^{-5}, 10^{-4}, 3{\cdot}10^{-4}, 10^{-3}, 3{\cdot}10^{-3}\}$.

The optimal shrinkage parameter is chosen to maximize accuracy subject to constraining unintended class collapse (at most 3% of classes merged). The Object-First check requires stronger covariance regularization (e.g., $3{\cdot}10^{-4}$ on CIFAR-100) because noisy cluster assignments introduce spurious scatter variance. Imposing baseline shrinkage on the Object-First detector causes severe accuracy degradation of 7.5 points on CIFAR-100 and escalates label inflation from 1.14 to 1.94.

B CATASTROPHIC FORGETTING AND NUMERICAL VERIFICATION

To confirm that deployment-time learning avoids catastrophic forgetting and maintains strict mathematical exactness, Table 4 compares deployed model states against models rebuilt from scratch on the final label multiset.

	Relative Covariance Gap $\|\Delta\mathbf{\Sigma}\|_F/\|\mathbf{\Sigma}\|_F$				Identical Predictions on Test Set (%)			
Deployment Agent	C100	TinyIN	IN-R	CUB	C100	TinyIN	IN-R	CUB
Term-First check (ours)	$6{\cdot}10^{-15}$	$2{\cdot}10^{-15}$	$4{\cdot}10^{-15}$	$2{\cdot}10^{-15}$	100.0	100.0	100.0	100.0
+ re-vote, exact moves	$6{\cdot}10^{-15}$	$8{\cdot}10^{-15}$	$4{\cdot}10^{-15}$	$2{\cdot}10^{-15}$	100.0	100.0	100.0	100.0
Learn now, exact moves	$7{\cdot}10^{-15}$	$9{\cdot}10^{-15}$	$4{\cdot}10^{-15}$	$2{\cdot}10^{-15}$	100.0	100.0	100.0	100.0
+ re-vote, add copies	$4{\cdot}10^{-3}$	$7{\cdot}10^{-3}$	10^{-3}	$2{\cdot}10^{-2}$	99.3	99.0	99.9	99.6
Learn now, add copies	$2{\cdot}10^{-2}$	$2{\cdot}10^{-2}$	10^{-2}	$8{\cdot}10^{-2}$	71.5	67.8	47.9	91.5

Table 4: Numerical exactness comparison between the online deployed model and a batch model reconstructed from scratch on the retained dataset (float64, Seed 0). Exact unlearning agrees with batch retraining to machine floating-point precision, whereas additive heuristics produce large structural errors and prediction discrepancies.

In double precision (float64), models employing exact unlearning match the from-scratch reconstruction within $9{\cdot}10^{-15}$ relative difference, with 100.0% identical predictions across all test instances. In contrast, the additive heuristic produces substantial covariance errors and alters prediction outputs.

C CHECKING THE OBJECT FIRST: ALGORITHMIC DETAILS

The Object-First detector reverses the operational hierarchy: it groups incoming observations based on visual feature clustering before assigning semantic class names. This section details its mathematical formulation.

C.1 VISUAL CLUSTERING VIA HOTELLING'S TWO-SAMPLE STATISTIC

The agent maintains visual clusters, each defined by sample count n_a, running sum $\mathbf{s}_a$, and class mean $\boldsymbol{\mu}_a = \mathbf{s}_a/\|\mathbf{s}_a\|_2$. To evaluate whether a new sample $\tilde{\mathbf{z}}$ or an existing visual group a should

merge with group b, we compute Hotelling's two-sample T^2 statistic under the shared precision matrix $\mathbf{\Lambda}$:

$$T^2(a,b) = \frac{n_a n_b}{n_a + n_b}(\boldsymbol{\mu}_a - \boldsymbol{\mu}_b)^\top \mathbf{\Lambda}(\boldsymbol{\mu}_a - \boldsymbol{\mu}_b). \tag{15}$$

The factor $\frac{n_a n_b}{n_a+n_b}$ accounts for sample support: large clusters must exhibit closer statistical proximity than small clusters to warrant merging, reflecting increased certainty in their mean estimates. To map $T^2(a,b)$ into a merge probability, we calibrate a logistic sigmoid model using validation splits prior to deployment:

$$P(\text{same visual class} \mid a,b) = \sigma\left(\alpha T^2(a,b) + \beta\right), \tag{16}$$

where $\alpha < 0$ and $\beta \in \mathbb{R}$ are calibrated parameters.

C.2 Decision Thresholds and Cluster Membership

To govern cluster updates, the similarity score is evaluated against three hierarchically calibrated thresholds: 1. $\tau_{\text{tentative}}$: A permissive threshold for merging into unconfirmed, tentative clusters, where false joins carry minimal risk. 2. τ_{settled}: A strict threshold for admitting new observations into established clusters, preventing distribution contamination. 3. τ_{cross}: A conservative threshold for merging two clusters that carry distinct established labels, ensuring that linguistic synonyms are merged only under overwhelming visual evidence.

C.3 Semantic Label Assignment and Reliability Weighting

Each visual cluster maintains candidate label assignments scored in log-odds. The log-odds score for assigning label c to cluster a incorporates:

$$\begin{aligned}\text{Score}(c \mid a) = &\sum_{u \in \mathcal{U}(a,c)} w_u + \gamma \log \frac{P_{\text{VLM}}(c \mid a)}{1 - P_{\text{VLM}}(c \mid a)} \\ &- \rho_1 \cdot \mathbb{I}[\text{cluster has alternative label}] - \rho_2 \cdot \mathbb{I}[c \in \text{OtherClusters}],\end{aligned} \tag{17}$$

where $\mathcal{U}(a,c)$ denotes the set of unique users who provided label c for cluster a, and user reliability weights w_u are derived from online accuracy estimates (Dawid & Skene, 1979):

$$w_u = \log \frac{\hat{p}_u}{1-\hat{p}_u} - \log \frac{p_0}{1-p_0}, \tag{18}$$

with p_0 representing baseline chance accuracy. The penalty terms ρ_1 and ρ_2 resolve competition among conflicting labels and discourage cluster duplication.

D Ablation Studies on the Object-First Check

Table 5 presents component ablations for the Object-First detector across all four benchmark datasets.

	CIFAR-100		TinyImageNet		ImageNet-R		CUB-200	
Ablation Variant	ΔA	ΔI	ΔA	ΔI	ΔA	ΔI	ΔA	ΔI
Remove VLM vote	−10.3	+1.44	−11.0	+1.58	−8.9	+0.99	−8.7	+1.23
Uncorrected reliability weights	−4.0	+0.20	−6.7	+0.30	−4.2	+0.30	+1.9	−0.00
No reliability weighting	−0.2	+0.03	−1.2	+0.20	−0.3	+0.02	+0.0	+0.00
Binary yes/no VLM verification	−0.8	+0.10	−1.9	+0.26	−2.2	+0.11	+1.6	−0.10

Table 5: Ablations of the Object-First detection strategy, reporting changes in matched accuracy (ΔA, percentage points) and label-space inflation (ΔI) relative to the full Object-First detector. Positive ΔA indicates improved accuracy.

The VLM verification vote represents the single most critical component across all benchmarks, removing it causes substantial accuracy drops on all datasets.

E EXPERIMENTAL DETAIL

User	Shown an eggplant, types	Right
Reliable ($\times 4$)	"eggplant"	90%
Sloppy ($\times 2$)	"eggplant" or a look-alike	70%
Synonym ($\times 2$)	"aubergine"	90%
Broader ($\times 1$)	"solanaceous vegetable"	90%
Foreign ($\times 1$)	"lex0042", unreadable by any model	90%
Adversarial ($\times 1$)	"apple", by a fixed wrong mapping	10%

Table 6: The default user crowd. Each detected sample is labelled once, by one user drawn at random.

Method	How it decides	Given "aubergine" for an eggplant
No curation	Accepts every term as a class term.	Creates a second class for the same vegetable.
Redundant majority	Asks two more users and votes.	Two more users agree with the term. Creates a second class.
VLM veto, given all terms	Lets VLM pick among the true class terms.	"aubergine" is not on the list; "eggplant" wins. No second class — but only because it was handed the answer key.
VLM veto	Lets VLM pick among the terms heard so far.	"aubergine" is now on the list and VLM likes it. Creates a second class.
Agreement gate	Learns only when two of three sources agree.	VLM confirms "aubergine". Creates a second class.
Term-first check	The agreement gate, but a new term is first put to a language model as a possible synonym of a known one.	"aubergine" is joined to "eggplant". **No second class.**
Object-first check	Groups images by appearance first, terms the groups afterwards.	The image joins the eggplant group; the term is joined later. **No second class.**

Table 7: One correct but unfamiliar label, passed through every method. Only the two detectors of Section 4 avoid a duplicate class without being given the answer key.

	Many-to-one accuracy (%)				Term recall			
Method	C100	TinyIN	IN-R	CUB	C100	TinyIN	IN-R	CUB
Clean oracle	88.1	84.9	78.1	83.7	1.00	1.00	1.00	1.00
No curation	86.6	83.8	74.4	80.2	1.00	1.00	1.00	0.99
VLM veto	84.6	82.7	75.4	71.5	1.00	1.00	1.00	0.90
Agreement gate	86.9	84.0	75.1	78.9	1.00	1.00	1.00	0.97
Term-first check	86.7	83.5	74.5	78.7	1.00	1.00	1.00	0.97
Object-first check	82.0	81.2	74.2	76.5	0.97	0.98	0.99	1.00

Table 8: Many-to-one accuracy lets several groups count for one class, so it does not charge for fragmentation; a method that splits classes into pure pieces scores well on it. Term recall is the share of true classes that some group terms, so below 1.00 means classes were merged away.

F REVISING DECISIONS THE AGENT MADE ITSELF

In Algorithm 1 the agent is one of the three voters, and it keeps changing: a vote cast at step 100 is one the step-2000 agent might not cast. Every R steps it can re-run the vote on every decision that depended on its own prediction, using the model as it is now:

- a *held-back* sample whose vote now agrees on a term is learned under that term;

- a *learned* sample whose vote now points to a different term is moved there, by removing it from its current class and learning it under the new one;
- a sample learned because the user and the vision–language model agreed, or because a person answered a question about it, is never revisited: no new evidence has arrived about it.

A learned sample is scored with itself left out of its own class, which the stored statistics allow exactly, since the mean without $\tilde{\mathbf{z}}$ is $(\mathbf{s}_k - \tilde{\mathbf{z}})/\|\mathbf{s}_k - \tilde{\mathbf{z}}\|$; otherwise it would always vote for the class it is already in. Of the three actions, only moving a learned sample needs unlearning. We therefore compare three ways of acting on a re-vote: releasing held-back samples only (+ *re-vote, release only*), also moving learned samples exactly (+ *re-vote, exact moves*), and moving them by adding a copy under the new class without removing the old one, which is all a learner without unlearning can do (+ *re-vote, add copies*). We also report *learn-now* agents, which hold nothing back: they learn every undecided sample at once under the user's term and rely on later corrections, applied exactly, by adding copies, or not at all. Table 2 repairs three of these agents at the end of deployment: + *re-vote, exact moves*, *Learn now, exact moves* and *Learn now, add copies*.

Table 9 re-runs the vote of Algorithm 1 every 100 stream steps on the decisions that used the agent's own prediction. The interval hardly matters: over 25, 50, 200 and 400 steps accuracy moves by at most 0.6 points on any dataset (Table 10).

Re-voting helps, but modestly, and releasing held-back samples accounts for all of it: adding exact moves on top changes nothing measurable (-0.0 points, $[-0.1, +0.1]$), because the re-vote moves very few learned samples, 66 per run on CIFAR-100 and 2 on ImageNet-R. An honest reading is that when a detector holds data back, the ability to forget is a safety property rather than an accuracy gain.

The picture changes when the agent stops holding data back and learns every undecided sample immediately under the user's term, relying on later corrections. Now corrections are many — 747 rank-one updates on CIFAR-100 — and how they are applied decides the outcome. Applying them exactly is worth $+1.6$ points ($[+1.0, +2.2]$) over adding the corrected copy without removing the wrong one, and $+2.0$ ($[+1.4, +2.7]$) over recording the correction and not acting on it. This is what the lower half of Table 9 shows: learning everything immediately and correcting exactly ends level with holding data back while using the whole stream, and without exact forgetting it falls behind. Doing the same corrections by retraining would cost, per correction, as many rank-one updates as the model holds samples.

Agent	C100	TinyIN	IN-R	CUB	All runs vs. row 1
Term-First check (ours)	76.1	73.3	68.9	77.3	—
+ re-vote, release only	76.9	73.5	69.2	77.9	$+0.5$ $[+0.2, +0.8]$
+ re-vote, exact moves	76.7	73.7	69.2	77.8	$+0.5$ $[+0.1, +0.8]$
+ re-vote, add copies	76.8	73.6	69.3	77.9	$+0.5$ $[+0.1, +0.9]$
Learn now, exact moves	77.0	73.8	67.3	77.6	$+0.0$ $[-0.6, +0.7]$
Learn now, add copies	74.8	71.3	66.2	77.1	-1.6 $[-2.2, -0.9]$
Learn now, ignore corrections	73.7	71.2	65.5	77.1	-2.0 $[-2.8, -1.2]$

Table 9: Matched accuracy (%) with the agent's own past decisions open to revision, and the paired difference from the Term-First check over all 3 seeds of every dataset. When decisions are few, exact forgetting changes little; when the agent learns everything immediately, it is worth over a point.

Re-vote every (stream steps)	25	50	200	400
CIFAR-100	76.8	76.9	77.3	77.3
TinyImageNet	73.6	74.0	73.5	73.4
ImageNet-R	69.0	69.1	69.2	69.1
CUB-200	77.6	77.6	77.6	77.6

Table 10: Matched accuracy (%) of the Term-First check with re-vote and exact moves, seed 0, as the re-vote interval changes.

G Model Recovery Trajectories Under Exact Unlearning

Figure 3 shows the model recovery trajectories under exact unlearning for an agent that initially learned all incoming data without prior curation (*No Curation*). As erroneous labels are unlearned and relocated to their correct classes in random order, accuracy converges monotonically to the Clean Oracle baseline. Because every single downdate produces an algebraically exact LDA solution, the classifier remains strictly valid and servable at every intermediate point.

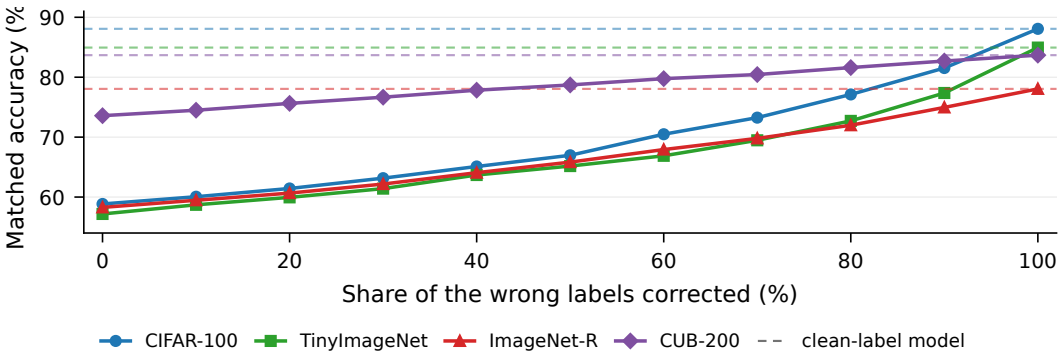


Figure 3: Model recovery trajectories under exact unlearning for an agent that initially learned all incoming data. As erroneous labels are unlearned and corrected in random order, accuracy converges monotonically to the Clean Oracle baseline (dashed). At every intermediate state, the model remains mathematically valid and servable.

H Computational Cost, Runtime Scaling, and Hardware Details

All experiments are conducted on a single NVIDIA A6000 GPU utilizing $\sim$7 GB of memory.

Per-Sample Correction vs. Full Retraining. Computationally, updating a sample takes milliseconds compared to full retraining from scratch on stored features, breaking even for up to hundreds of corrections. Each correction requires only an algebraic rank-one downdate of the scatter matrix to remove the sample from its corrupted class and a corresponding rank-one update to add it to the correct class, leaving all other class statistics untouched in $O(D^2)$ time.

Precision Matrix Inversion and Lazy Evaluation. During online streaming deployment and model repair, the regularized precision matrix $\mathbf{\Lambda} = ((1-\epsilon)\mathbf{\Sigma} + \epsilon\mathbf{I})^{-1}$ is not re-inverted after every single sample correction. Instead, it is lazily recomputed via Cholesky decomposition or refreshed via the Woodbury matrix identity only when the first downstream inference request arrives after a sequence of edits, caching the factorized state for all subsequent predictions. For batch operations (such as multi-sample relabeling or taxonomy merging), refreshing $\mathbf{\Lambda}$ via the Woodbury rank-four formula takes only 24 ms ($O(D^2)$), compared to 2551 ms for full $O(D^3)$ Cholesky factorization, offering over a $100\times$ speedup while matching the rebuilt model to machine floating-point precision.

The Cost of an Edit Does Not Grow with Its Size. Relabelling 1000 samples takes 87.8 ms as one edit and 61.5 s one sample at a time, the batch editing process is $700\times$ faster. Refreshing the precision matrix after a relabel by the Woodbury identity takes 24 ms, against 2551 ms for a fresh factorisation. Every edit agrees with the rebuilt model to $9.9{\cdot}10^{-15}$ relative covariance gap or better.