PARAMETER-EFFICIENT DATA-FREE UNTRAINING IN TEST-TIME ADAPTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

In real-world applications, online adaptation of a source-trained model to continually changing target domains is becoming increasingly common and important. This process is known as test-time adaptation (TTA) and is used in applications such as driving and aerial perception. This inevitably raises data-related risks: continually arriving data may contain corrupted samples or contaminated information, which can impair model reliability, as well as sensitive information, which can raise privacy risks. This creates an urgent need to remove the knowledge of problematic samples while preserving useful updates and source knowledge during online deployment, without access to historical raw inputs. To the best of our knowledge, existing machine unlearning methods do not address this problem. We first distinguish the two notions of untraining and unlearning, and address this need by proposing a novel untraining setting: Data-Free Untraining in TTA. For this setting, we propose a batch-level untraining method based on a parameter-efficient TTA structure that combines model-state storage, contrastive reconstruction, and selective untraining. We represent the unavailable requested data by reconstructed tokens and remove their knowledge via selective untraining while preserving the retained adaptation and source knowledge. We evaluate our method with different TTA methods on datasets commonly used in TTA. Our method outperforms all adapted MU baselines and maintains stable performance across successive untraining requests.

1 INTRODUCTION

As model sizes and training datasets grow, training data may contain sensitive, private or harmful content. Such data may need to be withdrawn after training to protect privacy or restore model reliability (Goel et al., 2024). The GDPR provides a right to erasure under specified conditions, while HIPAA Privacy Rule requires covered entities to safeguard health information against impermissible uses and disclosures (European Parliament and Council, 2016; HHS, 2025). While deleting these data is straightforward, removing their effect on the model's parameters is not, and retraining the model is often prohibitively expensive. To address this problem, many *machine unlearning* (MU) methods focus on removing the influence of a set of training data without full retraining



Figure 1: In TTA, some incoming batches contain problematic samples which request untraining.

What is TTA, and why does it need untraining? In the test-time adaptation (TTA) setting, a model is first trained on labeled source-domain data and then adapted online to the incoming target data (Wang et al., 2022). For example, a driving system may adapt to frames affected by sensor faults, while an aerial perception system may process images of restricted facilities. For personal data, residual training information also poses privacy risks, *e.g.,* membership inference can reveal whether a sample participated in training (Shokri et al., 2017). However, removing historical data influence during continual online adaptation has received less attention than offline removal.

Why untraining rather than unlearning? Existing MU works span two distinct problem formulations: untraining and unlearning, both referred to as "unlearning" in the literature. Triantafillou et al. (2026) define untraining as reversing the influence of a specified finite training set, whereas

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.

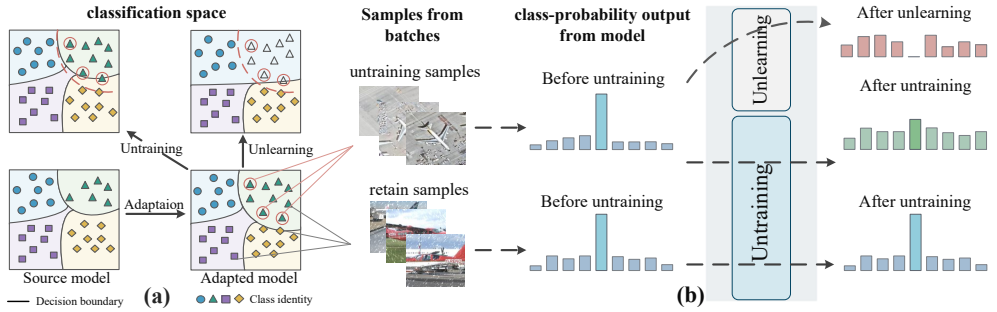


Figure 2: (a) Adaptation, untraining and unlearning in the classification space. The untraining request specifies the circled samples. (b) Representative class-probability outputs for untraining and retained TTA images before, after untraining, and after unlearning.

unlearning removes the broader distribution, concept, or behavior represented by that set. Figure 2 illustrates this difference in the classification space and the class-probability outputs. Untraining corrects the boundary changes associated with these untraining request samples unlearning instead removes broader knowledge of class. Our task exactly follows the untraining definition.

In this work, we propose a data-free untraining setting for continual TTA. Figure 1 illustrates this setting. Past target samples are often unavailable or difficult to retrieve, and untraining requests may arrive after further adaptation. Exact sample-level untraining is ideal, but unavailable without raw data because a TTA model is updated at the batch level. We therefore propose batch-level untraining that matches the unit of online adaptation. We consider untraining requests for one or more complete batches that contain problematic samples, even if only some samples in each batch are problematic. Our method can effectively remove the influence of problematic target samples without raw data while preserving retained adaptation and source knowledge.

Table 1: **Comparison of existing task settings.** Abbreviations: ON, online; DF, data-free; CD, cross-domain; SK, sample-wise knowledge; CK, category-wise knowledge.

Task	ON	DF	CD	SK	CK	Methods
TTA / CTTA	✓	×	✓	↑	↑	TENT, ViDA
Domain adaptation	×	×	✓	↑	↑	DANN, CDAN
Continual learning	✓	×	×	↑	↑	EWC, iCaRL
Conventional MU	×	×	×	↓	↓	SalUn, SCRUB
DF MU	×	✓	×	↓	↓	UISPS
CD MU	×	✓	✓	↓	↓	SCADA-UL
Untraining	×	×	×	↓	→	–
DF TTA untraining	✓	✓	✓	↓	→	This work

Table 1 summarizes the differences between our method and existing work. Existing MU methods typically perform offline removal using the untraining set, the retain set, or both (Foster et al., 2024). Data-free MU methods such as UISPS (Lian et al., 2025) avoid access to untraining data but perform category-level knowledge removal. Though SCADA-UL (Devalapally et al., 2026) supports removal during domain adaptation (including successive requests), it removes knowledge of source-exclusive classes rather than sample-wise knowledge. Within a continuous target stream, our method performs online removal of sample-wise knowledge without raw data while preserving category-wise knowledge shared with other samples.

Our method has three key components. The first *parameter-efficient TTA component* adapts only a small set of trainable or lightweight modules to incoming target batches. we use *model state* to represent the trainable parameter values. The second *contrastive reconstruction component* reconstructs feature tokens that serve as substitutes for unavailable historical inputs during untraining. The third *selective untraining component* uses the reconstructed tokens to perform effective untraining, while Fisher-weighted regularization and feature-space gradient projection preserve retained knowledge. Our main contributions are:

- We propose a novel setting, *i.e.,* Data-Free Untraining in TTA, and introduce Retain-Only Replay as an oracle that adapts on the historical stream after excluding the requested batches.
- We propose an untraining method to address this setting based on parameter-efficient TTA, combining model-state storage, contrastive reconstruction and selective untraining.
- We evaluate sensitive-data and contamination untraining across datasets based on two TTA methods, with adapted MU baselines, sequential untraining requests, and component ablations.

2 Related Work

Test-time adaptation. TTA adapts source-trained models to unlabeled test inputs. Entropy-based methods (Wang et al., 2021; Zhang et al., 2022) rely on predictions whose confidence may not imply reliability (Guo et al., 2017). Continual TTA stabilizes updates through averaging and restoration (Wang et al., 2022), sample selection and regularization (Niu et al., 2022; 2023; Lee et al., 2024), or normalization and memory (Gong et al., 2022; Yuan et al., 2023; Zhao et al., 2023). ViDA (Liu et al., 2024) separates domain information through adapters, while active TTA (Gui et al., 2024) acquires selected labels.

Machine unlearning and untraining. Existing MU covers both finite-set influence removal and broader knowledge removal (Liu et al., 2025). Triantafillou et al. (2026) distinguish untraining from unlearning. Following this distinction, we review these objectives separately, although both have commonly been studied under the term MU. For untraining, SISA (Bourtoule et al., 2021) reduces retraining cost through isolated shards and checkpoints, while certified removal (Guo et al., 2020) bounds distinguishability from retraining for linear classifiers. For unlearning, SCADA-UL (Devalapally et al., 2026) removes source-exclusive class knowledge during domain adaptation through adversarial generation and relabeling, including successive class-removal requests. Approximate removal techniques used across this literature include weight scrubbing, Fisher-based dampening, and saliency-guided updates (Golatkar et al., 2020; Foster et al., 2024; Fan et al., 2024), as well as distillation that balances forgetting against retained behavior (Kurmanji et al., 2023; Zhou et al., 2025).

Data-free methods and related mechanisms. Data-free methods use model-derived signals to replace unavailable training data. Proxy generation and inversion (Chen et al., 2019; Cai et al., 2020; Yin et al., 2020; Lian et al., 2025) capture class or distribution information without identifying historical batches within a class. Related approaches reuse stored updates (Wu et al., 2020; Graves et al., 2021) or preserve knowledge through distillation (Hinton et al., 2015), retained-data curvature approximation (Ahmed et al., 2025), or gradient constraints (Farajtabar et al., 2020; Saha et al., 2021). Update reuse incurs history storage, while DeltaGrad and distillation still require inputs.

3 Parameter-Efficient Data-Free Untraining in TTA

3.1 Problem Statement

Test-Time Adaptation. Let Θ_0 be the source model pre-trained on labeled source data with C classes. TTA deploys this model on unlabeled target batches $\{X_1, \ldots, X_T\}$, where t is the batch index, X_t is the batch observed at that step, and T is the total number of batches. We denote the full model after processing batch X_t as $\Theta_t = (\theta_0, \psi_t)$. The target distribution may differ from the source distribution and change over time in continual TTA (Wang et al., 2022; Liu et al., 2024).

The goal of TTA is to maintain accurate predictions under distribution shifts by updating the model online with unlabeled target batches sequentially. Given X_t, the base TTA rule $\mathcal{A}$ updates the full model as $\Theta_t = \mathcal{A}(\Theta_{t-1}, X_t)$. The updated model Θ_t predicts the labels of the samples in the current batch. We define $Y_t \subseteq \{1, \ldots, C\}$ as the set of distinct classes predicted for samples in X_t during this adaptation step. The model state ψ_t is carried forward to the next batch.

Data-free TTA untraining. We denote an untraining request by $q = (I, t_{\text{req}})$. It arrives while the model is processing $X_{t_{\text{req}}}$. I is a nonempty set of historical batch indices, *i.e.,* $I = \{t_1, \ldots, t_m\}, 1 \leq t_1 < \cdots < t_m < t_{\text{req}}$. The request specifies only batch indices and contains no samples, labels, or class identities. The requested untraining dataset is:

$$\mathcal{D}_u(q) = \bigcup_{t \in I} X_t = \bigcup_{t \in I} \bigcup_{x \in X_t} \{x\}. \tag{1}$$

Neither the requested batches nor the remaining historical inputs are available when untraining. Ideally, untraining yields a corrected model Θ_u that approximates the model obtained by replaying the historical stream without $\mathcal{D}_u(q)$:

$$\Theta_u \approx \Theta_{\text{replay}} \triangleq \mathcal{A}\big(\Theta_0, \{X_t\}_{t < t_{\text{req}},\, t \notin I}\big)\,. \tag{2}$$

We denote Θ_{replay} as the Retain-Only Replay oracle, which is used only for evaluation and is not available to our method. Recognition of samples in $\mathcal{D}_u(q)$ can persist through knowledge shared with retained samples. After untraining, we continue the adaptation procedure from the corrected model Θ_u based on $X_{t_{\text{req}}}$, *i.e.,* $\Theta_{t_{\text{req}}} = \mathcal{A}(\Theta_u, X_{t_{\text{req}}})$.

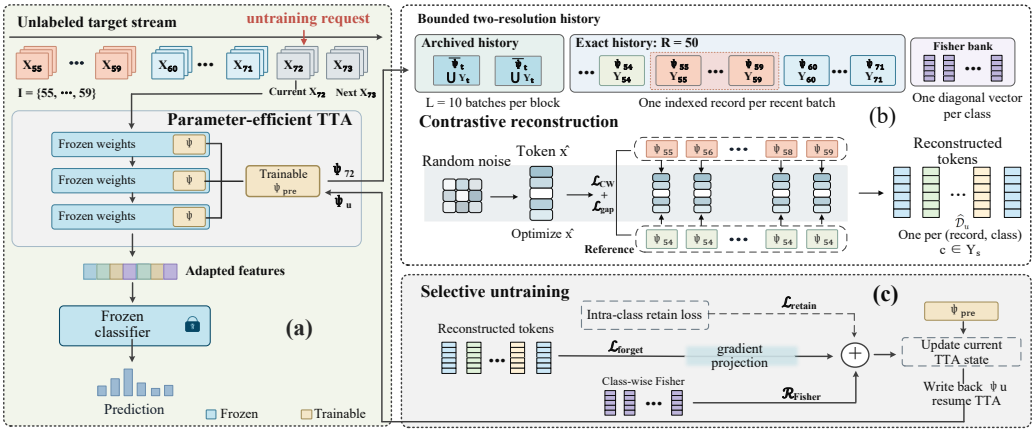


Figure 3: Overview of our method: (a) parameter-efficient TTA and record storage; (b) contrastive reconstruction of untraining tokens; and (c) selective untraining.

3.2 Proposed Methodology

Overview of our method. Figure 3 summarizes our framework. Parameter-efficient TTA updates the model state ψ_t while keeping the remaining parameters frozen. We maintain a storage-efficient model-state history containing batch indices, model states, and predicted class sets, alongside a class-wise Fisher bank of parameter-importance scores. When a request arrives, contrastive reconstruction uses the relevant history and a reference model state to generate untraining tokens. Selective untraining uses these tokens to correct the current model state, with Fisher-weighted regularization and feature-space gradient projection to limit changes that affect retained predictions. The corrected model then resumes TTA. Appendix B.1 gives the complete procedure in Algorithm B.1.

Parameter-efficient TTA. Online adaptation must adapt efficiently to target data while preserving source knowledge. We address this by updating selected parameters or lightweight modules rather than the full network. Following TENT (Wang et al., 2021), our Vision Transformer (ViT) implementation adapts only the scale and shift parameters of LayerNorm layers. The remaining backbone parameters and the classifier are frozen. Following ViDA (Liu et al., 2024), we alternatively insert low- and high-rank adapters alongside frozen operations within network layers. We build on these two parameter-efficient TTA architectures. The model state ψ_t contains the trainable parameters that learn feature adjustments from the incoming target batches. Their changes reflect adaptation to target data. At batch t, the full model is $\Theta_t = (\theta_0, \psi_t)$ with $|\psi_t| \ll |\theta_0|$, where the bars here denote the number of parameters. For an ordinary batch X_t, a base TTA step updates the model state as

$$\psi_t \leftarrow \psi_{t-1} - \eta_{\text{tta}} \nabla_{\psi} \ell_{\text{base}}(\Theta_{t-1}, X_t), \tag{3}$$

where ℓ_{base} is the objective of the chosen TTA method and $\Theta_{t-1} = (\theta_0, \psi_{t-1})$. We use ψ_{pre} to denote the model state immediately before an untraining update and ψ_u to denote the corrected model state.

Storage-efficient model-state history. When untraining, inputs are no longer available. We therefore store the model states and predicted class sets for reconstruction, together with a separate class-wise Fisher bank for parameter protection. As shown in Figure 3, the history is organized as *records*. We index each record by s and represent its three elements as $(\mathcal{B}_s, \psi_s, \mathcal{Y}_s)$, where $\mathcal{B}_s$ is the set of represented batch indices, ψ_s is the stored model state, and $\mathcal{Y}_s$ is the stored predicted-class set. The history contains exact records for recent batches and archived records for older batches. For an exact record representing batch t, $\mathcal{B}_s = \{t\}$, $\psi_s = \psi_t$, and $\mathcal{Y}_s = Y_t$; it retains the model state and predicted classes for each batch. We keep at most R exact records (*e.g.,* $R = 50$ in Fig. 3). Each group of L consecutive older batches forms a *block*. We replace its L exact records with one archived record and retain at most H archived records (*e.g.,* $L = 10$ and $H = 2$ in Fig. 3). For an archived record, $\mathcal{B}_s$ is its block, with

$$\psi_s = \tfrac{1}{L} \textstyle\sum_{t \in \mathcal{B}_s} \psi_t, \qquad |\mathcal{B}_s| = L, \qquad \mathcal{Y}_s = \bigcup_{t \in \mathcal{B}_s} Y_t. \tag{4}$$

When the archive is full, the oldest archived record is discarded. The separate class-wise Fisher bank holds at most C vectors over the trainable parameters. With R, L, and H fixed, storage does not grow with the stream length.

Class-wise Fisher bank. We therefore maintain a class-wise Fisher importance score vector over the trainable TTA parameters. Each vector is derived from squared loss gradients and assigns an importance score to every trainable parameter for its class. Let $F_{t,c} \in \mathbb{R}^{|\psi_t|}$ be the importance score vector for class c after batch t. We update it as

$$F_{t,c} = \beta_F F_{t-1,c} + (1 - \beta_F) \left(\frac{\partial \ell_{t,c}}{\partial \psi_t} \right)^2, \tag{5}$$

Here, $\ell_{t,c}$ is the adaptation loss averaged over samples in X_t that are used by the base TTA update and predicted as class c. The vector is initialized to be zero and remains unchanged when no reliable (confidence higher than a certain threshold) class-c sample is present. Each entry measures the importance of one trainable parameter, as illustrated in the class-wise Fisher bank of Figure 3. Appendix A.1 provides further details.

3.3 Contrastive Reconstruction of Untraining Tokens

Untraining needs inputs, but the requested batches are unavailable in our data-free setting. We therefore reconstruct feature tokens that act as substitutes in the untraining loss. Our reconstruction operates in feature space.

Reference model state. Token reconstruction should capture what the model learned from the requested batches. For each relevant record s, we therefore compare the class scores produced by its stored model state ψ_s with those produced by a reference model state ψ_{ref} captured before the earliest requested batch. This reference model state has not been adapted to any requested batch. When untraining covers batches in an archived record, we use ψ_s from the eligible archived record whose block ends closest to and before that batch (*e.g.,* ψ_{54} is the reference model state in Figure 3).

Reconstructing untraining tokens. We reconstruct one token $\hat{x}_{s,c}$ for each distinct record s with $\mathcal{B}_s \cap I \neq \varnothing$ and each class c in its stored set $\mathcal{Y}_s$. For an archived record, reconstruction uses the entire block's class union, including when only part of the block is requested. The classes used for reconstruction are therefore $Y_I = \bigcup_{s:\, \mathcal{B}_s \cap I \neq \varnothing} \mathcal{Y}_s$. Let $z_j(\psi_s; \hat{x})$ be the class-j logit, the score before probability normalization, produced from token $\hat{x}$ with frozen weights θ_0 implicit. Writing $z_j^s = z_j(\psi_s; \hat{x})$ and $z_j^{\text{ref}} = z_j(\psi_{\text{ref}}; \hat{x})$, we optimize

$$\hat{x}^\star_{s,c} = \arg\min_{\hat{x}} \Big[\underbrace{\max(0, \max_{j \neq c} z_j^s - z_c^s + \kappa)}_{\mathcal{L}_{\text{CW}}(\hat{x})} + \alpha_{\text{gap}} \underbrace{(z_c^{\text{ref}} - z_c^s)}_{\mathcal{L}_{\text{gap}}(\hat{x})} + \beta_{\text{norm}} \underbrace{\|\hat{x}\|_2^2}_{\text{norm penalty}} \Big]. \tag{6}$$

The Carlini–Wagner (C&W)-style margin loss (Carlini & Wagner, 2017) $\mathcal{L}_{\text{CW}}$ encourages the class-c score to exceed competing class scores by $\kappa \geq 0$, where j ranges over the C classes. The gap loss $\mathcal{L}_{\text{gap}}$ favors a higher class-c score under the retrieved model state than under the reference model state. The resulting tokens approximate the requested influence through changes in class responses between the stored and reference model states. Collecting the optimized tokens, denoted by $\star$, gives

$$\widehat{\mathcal{D}}_u = \left\{ (\hat{x}^\star_{s,c}, c) \;\middle|\; \mathcal{B}_s \cap I \neq \varnothing,\ c \in \mathcal{Y}_s \right\}. \tag{7}$$

Each element pairs an optimized feature token with the class $c \in \mathcal{Y}_s$ used to reconstruct it, matching the reconstructed-token columns in Figure 3.

3.4 Selective Untraining

Updating the model based only on reconstructed tokens would affect retained knowledge. We therefore combine a token-based untraining loss with Fisher-weighted regularization and feature-space gradient projection to perform effective untraining while preserving retained knowledge. An optional intra-class retain loss is used to preserve same-class knowledge when untraining and retained features substantially overlap.

Token-based untraining loss. Each pair $(\hat{x}, c) \in \widehat{\mathcal{D}}_u$ consists of a feature token reconstructed from a record with $\mathcal{B}_s \cap I \neq \varnothing$ and the class $c \in \mathcal{Y}_s$ used for reconstruction. To reduce the influence of the requested batches, we penalize positive alignment between each reconstructed token's adapted feature and its class classifier vector. Let $h_\psi(\hat{x})$ be the final adapted feature obtained by passing token $\hat{x}$ through the remaining network layers. The frozen classifier vector W_c specifies the feature direction used to score class c. We penalize positive cosine similarity between $h_\psi(\hat{x})$ and W_c:

$$\mathcal{L}_{\text{forget}} = \tfrac{1}{|\widehat{\mathcal{D}}_u|} \textstyle\sum_{(\hat{x},c) \in \widehat{\mathcal{D}}_u} \max(0, \rho(h_\psi(\hat{x}), W_c)), \tag{8}$$

where $\rho(h, W_c) = h^\top W_c/(\|h\|_2\|W_c\|_2)$ for nonzero vectors. The loss reduces positive feature–classifier alignment and is zero when that alignment is nonpositive.

Fisher-weighted regularization. To preserve retained knowledge, we use class-wise Fisher importance score vectors to penalize changes to important parameters, particularly those supporting classes outside Y_I. When an untraining request arrives, let F_c be the latest Fisher-bank vector for class c, and $\mathcal{C}_F$ the classes with available score vectors. We average them over classes outside and inside Y_I:

$$F_{\text{retain}} = \frac{\sum_{c\in\mathcal{C}_F\setminus Y_I} F_c}{|\mathcal{C}_F\setminus Y_I|}, \qquad F_{\text{untrain}} = \frac{\sum_{c\in\mathcal{C}_F\cap Y_I} F_c}{|\mathcal{C}_F\cap Y_I|}. \tag{9}$$

The two vectors measure each trainable parameter's importance to the respective class groups. If either class group is empty, its mean vector is defined as zero. We combine the importance vectors into weights $\widehat{F}$ and penalize changes in ψ using elastic weight consolidation (EWC) (Kirkpatrick et al., 2017):

$$\widehat{F} = F_{\text{retain}} + \alpha_F F_{\text{untrain}}, \qquad \mathcal{R}_{\text{Fisher}} = \textstyle\sum_j \widehat{F}_j(\psi_j - \psi_{\text{pre},j})^2. \tag{10}$$

Here, j indexes trainable parameters. Choosing $0 \le \alpha_F < 1$ reduces the penalty from F_{untrain}, allowing greater changes to parameters mainly important to classes in Y_I. The penalty from F_{retain} remains unchanged, preserving constraints on parameters important to other classes, including shared parameters. We combine the untraining loss with this penalty on parameter changes:

$$\mathcal{L}_{\text{total}} = \lambda_f\mathcal{L}_{\text{forget}} + \lambda_e\mathcal{R}_{\text{Fisher}}. \tag{11}$$

Feature-space gradient projection. Fisher-weighted regularization limits changes in parameter values but does not directly constrain how an untraining update changes class features. To reduce interference with shared class knowledge, we remove the component of the untraining feature gradient along the frozen class-c classifier direction. This direction is the classifier weight vector W_c. For token $(\hat{x}, c)$, let $G_{\hat{x}} = \partial\mathcal{L}_{\text{forget}}/\partial h_\psi(\hat{x})$ denote its gradient with respect to the adapted feature. We compute the projected feature gradient as

$$G_{\hat{x}}^{\perp} = G_{\hat{x}} - \frac{G_{\hat{x}}^\top W_c}{\max(\|W_c\|_2^2,\epsilon)} W_c. \tag{12}$$

When $\|W_c\|_2^2 \ge \epsilon > 0$, $\perp$ denotes projection onto the subspace orthogonal to W_c. Removing $G_{\hat{x}} - G_{\hat{x}}^{\perp}$ along W_c leaves the orthogonal feature-gradient component for backpropagation; Appendix A.2 provides the geometric illustration (Figure A.2) and a constrained-optimization derivation. We backpropagate the projected feature gradient through h_ψ to obtain the parameter gradient $g_{\text{forget}}^{\perp}$:

$$g = \lambda_f g_{\text{forget}}^{\perp} + \lambda_e\nabla_\psi\mathcal{R}_{\text{Fisher}}. \tag{13}$$

Here, g is the combined gradient used to update the trainable parameters.

Optional intra-class retain loss. When untraining and retained data substantially overlap in feature space, they may be visually similar.We therefore restrict retain-token reconstruction to the classes involved in the current untraining request: we reconstruct one retain token $\hat{x}_r$ for each class $r \in Y_I$. Starting from random noise, we optimize the retain token using the fixed reference model state ψ_{ref}:

$$\hat{x}_r^\star = \arg\min_{\hat{x}} \ \max(0, \max_{j\ne r} z_j(\psi_{\text{ref}};\hat{x}) - z_r(\psi_{\text{ref}};\hat{x}) + \kappa) + \beta_{\text{norm}}\|\hat{x}\|_2^2. \tag{14}$$

This objective uses the class-margin term and norm penalty from Eq. (6), but omits the gap loss. The resulting token–class pairs form the retain set $\widehat{\mathcal{D}}_r = \{(\hat{x}_r^\star, r) : r \in Y_I\}$. We preserve their feature–classifier alignment with

$$\mathcal{L}_{\text{retain}} = -\tfrac{1}{|\widehat{\mathcal{D}}_r|}\textstyle\sum_{(\hat{x}_r)\in\widehat{\mathcal{D}}_r}\max(0, \rho(h_\psi(\hat{x}_r), W_r)), \tag{15}$$

where $\hat{x}_r$ is a retain feature token and r is its reconstruction class. Minimizing this loss strengthens positive feature–classifier alignment. Combining the untraining loss, Fisher-weighted regularization, and intra-class retain loss gives the full objective:

$$\mathcal{L}_{\text{total}} = \lambda_f\mathcal{L}_{\text{forget}} + \lambda_e\mathcal{R}_{\text{Fisher}} + \lambda_p\mathcal{L}_{\text{retain}}. \tag{16}$$

During optimization, only the feature gradient of $\mathcal{L}_{\text{forget}}$ is projected. However, reconstructing these additional retain tokens incurs extra computation. In experiments, we find that this loss can be omitted in some scenarios while maintaining effective untraining and retention. We therefore configure its use in advance according to the application scenario. For example, for untraining of corrupted or contaminated data, our experiments demonstrate effective untraining and retention with $\lambda_p = 0$, avoiding retain-token reconstruction and recovering Eq. (11).

4 EXPERIMENTS

We evaluate three aspects of data-free TTA untraining: correction and retention relative to MU baselines, behavior under successive requests, and the contributions of correction components and reconstructed inputs.

Datasets. We use ImageNet (Deng et al., 2009) and CIFAR-100 (Krizhevsky, 2009) as source domains, with 1,000 and 100 classes, respectively. Their corrupted counterparts, ImageNet-C and CIFAR-100-C (Hendrycks & Dietterich, 2019), provide target streams with weather, noise, blur, and digital corruptions.

Baselines. We adapt SCADA-UL-style (Devalapally et al., 2026), DELETE (Zhou et al., 2025), SSD (Foster et al., 2024), SalUn (Fan et al., 2024), and SCRUB (Kurmanji et al., 2023) to the TTA parameter subspace; PEFT denotes parameter-efficient fine-tuning. No-Untrain keeps the adapted model, Source Reset restores the source model, and Clean Rollback restores a state before the requested batches. Retain-Only Replay (Oracle) reruns TTA from the source model up to request arrival, preserving stream order and excluding all requested batches in full (Eq. (2)).

Experimental Setup. We use ViT-B/16 (Dosovitskiy et al., 2021) with ViDA (Liu et al., 2024) and TENT (Wang et al., 2021). Requests specify historical batch indices. Adapted MU methods share the pre-correction state and trainable parameter subspace, using forget proxies and current-batch retain inputs. We evaluate immediately after correction, then resume TTA. Implementation and baseline adaptation details are in Appendix D.

Evaluation Metrics. F-Conf is mean confidence on requested samples using their pre-untraining predicted classes; CF-Gap is its absolute difference from Replay. SC-Acc and R-Acc measure same-class and retained-stream accuracy. Headers CF, SC, and R abbreviate these metrics. MIA-Gap is the absolute difference from Replay in mean membership probability on held-out requested members, using a fixed entropy attack calibrated on true members and nonmembers. Lower gaps and higher accuracies are better; the protocol is in Appendix D.4.

For the joint assessment of correction and retention, we report a composite Score on a 0–100 scale:

$$S = 25\big[(1 - \text{CF-Gap}) + (1 - \text{MIA-Gap}) + \text{SC-Acc} + \text{R-Acc}\big]. \tag{17}$$

Results with subscripts report means over five paired seeds and sample standard deviations.

Table 2: **Single untraining request comparison with ViDA on ImageNet-C and CIFAR-100-C.** Bold marks the best adapted MU result; reset, rollback, and Replay are controls.

ImageNet-C

Method	F-Conf	CF ↓	SC ↑	R ↑	MIA ↓	Score ↑
No-Untrain	$.878_{\pm.001}$	$.190_{\pm.005}$	$.938_{\pm.005}$	$.750_{\pm.011}$	$.221_{\pm.011}$	$81.93_{\pm.10}$
Source Reset	$.689_{\pm.005}$	$.001_{\pm.001}$	$.855_{\pm.010}$	$.683_{\pm.010}$	$.000_{\pm.000}$	$88.43_{\pm.11}$
Clean Rollback	$.681_{\pm.007}$	$.007_{\pm.007}$	$.836_{\pm.005}$	$.715_{\pm.007}$	$.004_{\pm.004}$	$88.50_{\pm.53}$
SCADA-UL-style	$.469_{\pm.005}$	$.219_{\pm.008}$	$.438_{\pm.003}$	$.734_{\pm.005}$	$.024_{\pm.004}$	$73.23_{\pm.41}$
DELETE-PEFT	$.632_{\pm.002}$	$.056_{\pm.006}$	$.562_{\pm.015}$	$.650_{\pm.007}$	$.024_{\pm.002}$	$78.30_{\pm.32}$
SSD-PEFT	$.878_{\pm.004}$	$.190_{\pm.006}$	$\mathbf{.938}_{\pm.015}$	$\mathbf{.750}_{\pm.004}$	$.221_{\pm.010}$	$81.93_{\pm.22}$
SalUn-PEFT	$.564_{\pm.008}$	$.124_{\pm.011}$	$.438_{\pm.015}$	$.725_{\pm.009}$	$.053_{\pm.009}$	$74.65_{\pm.55}$
SCRUB-PEFT	$.568_{\pm.015}$	$.120_{\pm.014}$	$.500_{\pm.001}$	$.722_{\pm.010}$	$\mathbf{.003}_{\pm.003}$	$77.48_{\pm.20}$
Ours	$.691_{\pm.007}$	$\mathbf{.003}_{\pm.003}$	$.875_{\pm.010}$	$\mathbf{.750}_{\pm.005}$	$.028_{\pm.013}$	$\mathbf{89.85}_{\pm.38}$
Retain-Only Replay	$.688_{\pm.004}$	$.000_{\pm.000}$	$.938_{\pm.007}$	$.750_{\pm.014}$	$.000_{\pm.000}$	$92.20_{\pm.25}$

CIFAR-100-C

Method	F-Conf	CF ↓	SC ↑	R ↑	MIA ↓	Score ↑
No-Untrain	$.918_{\pm.003}$	$.099_{\pm.010}$	$.828_{\pm.007}$	$.781_{\pm.004}$	$.163_{\pm.002}$	$83.68_{\pm.57}$
Source Reset	$.818_{\pm.012}$	$.001_{\pm.001}$	$.748_{\pm.005}$	$.751_{\pm.009}$	$.001_{\pm.001}$	$87.43_{\pm.40}$
Clean Rollback	$.819_{\pm.013}$	$.000_{\pm.000}$	$.768_{\pm.013}$	$.751_{\pm.009}$	$.000_{\pm.000}$	$87.98_{\pm.12}$
SCADA-UL-style	$.018_{\pm.012}$	$.801_{\pm.013}$	$.195_{\pm.006}$	$.750_{\pm.009}$	$.278_{\pm.010}$	$46.65_{\pm.21}$
DELETE-PEFT	$.000_{\pm.000}$	$.819_{\pm.013}$	$.000_{\pm.000}$	$.633_{\pm.007}$	$.241_{\pm.012}$	$39.33_{\pm.12}$
SSD-PEFT	$.918_{\pm.007}$	$.099_{\pm.014}$	$.728_{\pm.010}$	$.751_{\pm.009}$	$\mathbf{.063}_{\pm.011}$	$82.93_{\pm.59}$
SalUn-PEFT	$.002_{\pm.002}$	$.817_{\pm.015}$	$.109_{\pm.002}$	$.727_{\pm.001}$	$.279_{\pm.011}$	$43.50_{\pm.21}$
SCRUB-PEFT	$.328_{\pm.010}$	$.491_{\pm.011}$	$.563_{\pm.012}$	$.734_{\pm.003}$	$.247_{\pm.011}$	$63.98_{\pm.36}$
Ours	$.766_{\pm.004}$	$\mathbf{.053}_{\pm.015}$	$\mathbf{.782}_{\pm.014}$	$\mathbf{.781}_{\pm.004}$	$.108_{\pm.010}$	$\mathbf{85.05}_{\pm.47}$
Retain-Only Replay	$.819_{\pm.013}$	$.000_{\pm.000}$	$.828_{\pm.005}$	$.781_{\pm.010}$	$.000_{\pm.000}$	$90.23_{\pm.37}$

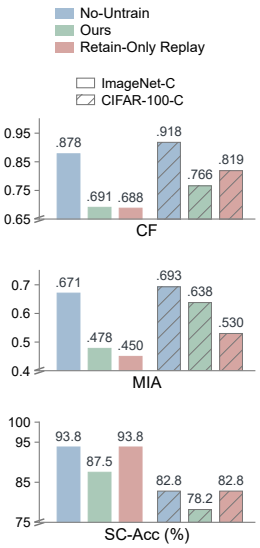


Figure 4: Untraining request outputs and retention. CF/MIA show raw confidence/attack scores; SC-Acc is accuracy. Linear axes start at 0.65/0.40/75%, respectively.

4.1 Untraining Effectiveness

We test whether correction removes sensitive-data and corrupted-data influence while preserving retained knowledge. We use ARJ21 and Boeing747 images from FAIR1M (Sun et al., 2022) for ImageNet-C and synthetically perturbed stream images for CIFAR-100-C. Based on normal TTA, we insert dozens of batches mixing these inputs with normal samples, then request removal of the inserted batches with $\lambda_p = 0.25$. Appendices D.1 and D.3 specify the construction and details.

Table 2 shows our lowest CF-Gap among adapted MU methods on both datasets, with retained-stream accuracy matching Replay. On ImageNet-C, SCADA-UL-style, SalUn-PEFT, and SCRUB-PEFT suppress confidence below Replay while reducing same-class accuracy; SSD-PEFT preserves same-class accuracy but leaves CF-Gap unchanged from No-Untrain. Our method improves both confidence agreement and same-class retention over the former group and achieves the highest adapted-method SC-Acc on CIFAR-100-C. Although below No-Untrain in SC-Acc (Figure 4), it retains 1.4 percentage points more SC-Acc and 3.0 points more R-Acc than Clean Rollback on CIFAR-100-C, where rollback has closer confidence agreement. These results show that our method more closely matches Replay while preserving useful same-class and retained-stream knowledge.

4.2 Untraining under Continued Adaptation

We test our method's performance across successive requests and different TTA parameterizations.

Sequential untraining. We compare against adapted MU baselines over three successive sensitive-data untraining requests on ImageNet-C, using mixed batches as in Table 2 and additional FAIR1M categories such as warships and sports cars (1–4 classes per batch). All methods update the same ViDA parameter subspace, with $\lambda_p = 0.25$ for Ours. Each corrected state is evaluated against cumulative Retain-Only Replay before adaptation resumes.

Table 3: **Sequential-untraining comparison on ImageNet-C with ViDA.** Values are fractions; bold CF/SC entries mark the best adapted MU result.

	First request				Second request				Third request			
Method	F-Conf	CF ↓	SC ↑	R ↑	F-Conf	CF ↓	SC ↑	R ↑	F-Conf	CF ↓	SC ↑	R ↑
No-Untrain	.878	.190	.938	.750	.036	.446	.688	.688	.061	.501	.812	.875
Source Reset	.689	.001	.855	.683	.480	.002	.812	.625	.361	.201	.812	.875
Clean Rollback	.681	.007	.836	.715	.500	.018	.812	.625	.384	.178	.875	.875
SCADA-UL-style	.469	.219	.438	.734	.224	.258	.562	.750	.025	.537	.125	.875
DELETE-PEFT	.632	.056	.562	.650	.772	.290	.750	.750	.721	.159	.750	.938
SSD-PEFT	.878	.190	**.938**	.750	.368	.114	.750	.750	.075	.487	.750	.875
SalUn-PEFT	.564	.124	.438	.725	.373	.109	.688	.750	.232	.330	.562	.875
SCRUB-PEFT	.568	.120	.500	.722	.306	.176	.688	.688	.076	.486	.750	.938
Ours	.691	**.003**	.875	.750	.589	**.107**	**.812**	.625	.684	**.122**	**.812**	.812
Retain-Only Replay	.688	.000	.938	.750	.482	.000	.812	.625	.562	.000	.812	.875

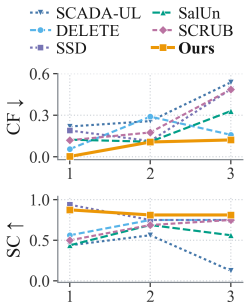


Figure 5: Untraining and same-class retention performance across requests. Each line tracks one adapted MU method or Ours over three requests.

Table 3 and Figure 5 show the lowest CF-Gap among adapted MU methods at every request and same-class accuracy matching Replay at the last two requests. At the third, SC-Acc is 0.812 versus 0.125–0.750 for these baselines, while R-Acc is lower (0.812 versus 0.875–0.938). Appendix C.1 covers cross-dataset comparisons and earlier-request persistence. These results show that our method maintains its advantage over adapted MU baselines across successive requests.

Repeated corrupted-data requests. We test five successive corrupted-data untraining requests with ViDA on ImageNet-C and transfer to single-model TENT on both benchmarks. Requested batches contain only synthetic perturbed images, with retain-token reconstruction disabled ($\lambda_p = 0$). Appendix D gives the construction and details.

Table 4: **Corrupted-data untraining requests based on TENT.** $\lambda_p = 0$.

Dataset	Request	F-Conf	CF↓	R↑
ImageNet-C	First	$.712_{\pm.010}$	$.038_{\pm.011}$	$.695_{\pm.014}$
	Fifth	$.840_{\pm.006}$	$.239_{\pm.013}$	$.711_{\pm.012}$
CIFAR-100-C	First	$.882_{\pm.003}$	$.062_{\pm.012}$	$.825_{\pm.006}$
	Third	$.800_{\pm.014}$	$.012_{\pm.005}$	$.688_{\pm.001}$

Table 5: **Continued corrupted-data untraining on ImageNet-C with ViDA.** $\lambda_p = 0$.

Request	F-Conf	CF↓	SC↑	R↑	MIA↓
First	$.583_{\pm.001}$	$.055_{\pm.011}$	$.938_{\pm.002}$	$.750_{\pm.002}$	$.033_{\pm.011}$
Third	$.899_{\pm.013}$	$.192_{\pm.001}$	$.875_{\pm.011}$	$.875_{\pm.012}$	$.198_{\pm.014}$
Fifth	$.674_{\pm.010}$	$.139_{\pm.007}$	$.688_{\pm.003}$	$.938_{\pm.009}$	$.070_{\pm.009}$

ViDA. Table 5 reports the performance after the first, third, and fifth untraining requests. R-Acc rises from 75.0% to 93.8%, while SC-Acc declines from 93.8% to 68.8%. CF-Gap rises from 0.055 to 0.192, then falls to 0.139. These results show that our method can interleave repeated corrupted-data correction with continued ViDA adaptation without retain-token reconstruction.

TENT. Table 4 shows ImageNet-C R-Acc near 70% as CF-Gap increases from 0.038 to 0.239. On CIFAR-100-C, CF-Gap improves from 0.062 to 0.012 while R-Acc falls from 82.5% to 68.8%. These results extend the correction procedure to single-model TENT under successive corrupted-data requests.

4.3 Ablation Study

We test the roles of correction components and reconstructed inputs.

Component ablations. For the first ImageNet-C sensitive-data untraining request with ViDA, we remove one component at a time (Table 6), holding the pre-correction state, evaluation samples, and update budget fixed. Removing Fisher weights retains an unweighted quadratic penalty; removing EWC eliminates it. Appendix D.3 specifies each configuration.

Without the reconstruction reference, confidence remains above No-Untrain and CF-Gap increases from 0.003 to 0.303. These results show that the reference state makes reconstructed tokens more effective for correcting adaptation-induced confidence changes.

Table 6: **Component ablations on ImageNet-C with ViDA.** Bold marks the best result; output gaps and retained accuracies use the same Replay reference.

Variant	F-Conf	CF↓	SC↑	R↑	MIA↓	Score↑
Full method	.691$_{\pm.007}$	**.003**$_{\pm.003}$	.875$_{\pm.010}$	**.750**$_{\pm.005}$	**.028**$_{\pm.013}$	**89.85**$_{\pm.38}$
w/o reference model state	.991$_{\pm.004}$	.303$_{\pm.000}$	**.938**$_{\pm.015}$	.725$_{\pm.009}$	.304$_{\pm.007}$	76.40$_{\pm.47}$
w/o Fisher weights	.798$_{\pm.015}$	.110$_{\pm.011}$	.875$_{\pm.001}$	.650$_{\pm.008}$	.151$_{\pm.015}$	81.60$_{\pm.51}$
w/o EWC	.215$_{\pm.001}$	.473$_{\pm.003}$	**.938**$_{\pm.005}$	.438$_{\pm.007}$	.208$_{\pm.008}$	67.38$_{\pm.30}$
w/o orthogonal projection	.006$_{\pm.002}$	.682$_{\pm.002}$	.000$_{\pm.000}$	.685$_{\pm.014}$	.295$_{\pm.003}$	42.70$_{\pm.36}$
w/o retain loss	.247$_{\pm.011}$	.441$_{\pm.007}$	.312$_{\pm.012}$	**.750**$_{\pm.011}$	.218$_{\pm.003}$	60.08$_{\pm.45}$

Removing Fisher weights reduces R-Acc from 75.0% to 65.0% without changing SC-Acc; removing EWC lowers R-Acc further to 43.8% and excessively suppresses confidence. This result means that Fisher weighting improves the retained-stream protection provided by the quadratic regularizer.

Without projection, same-class accuracy collapses; without the retain loss, it falls to 31.2% while R-Acc is unchanged. These results show that gradient projection and retain loss guidance complement parameter protection to balance correction with retained knowledge when untraining and retained data substantially overlap in feature space.

Reconstructed versus real inputs. We test whether reconstructed tokens substitute for historical raw inputs in the first ImageNet-C sensitive-data untraining request with ViDA (Table 7). The real-input variant uses all images in the requested batches instead of untraining tokens. Both variants share reconstructed retain tokens, the pre-correction state, update budget, and loss weights, including $\lambda_p = 0.25\lambda_f$.

Tokens reduce SC-Acc from 88.2% to 87.5% relative to real images, with similar CF-Gap of 0.002 and 0.003 and MIA gaps of 0.028 and 0.030. Both variants retain 75.0% R-Acc, matching Replay. The resulting Score gap is only 0.15. These results show that reconstructed tokens are a suitable substitute for raw data, supporting effective untraining without historical raw images.

Table 7: **Reconstructed versus real untraining inputs.** First-request ImageNet-C setting of Table 2.

Input	F-Conf	CF↓	SC↑	R↑	MIA↓	Score↑
Tokens	.691$_{\pm.007}$	.003$_{\pm.003}$	.875$_{\pm.010}$	.750$_{\pm.005}$	.028$_{\pm.013}$	89.85$_{\pm.38}$
Real images	.686$_{\pm.014}$	.002$_{\pm.010}$	.882$_{\pm.003}$	.750$_{\pm.002}$	.030$_{\pm.001}$	90.00$_{\pm.27}$
Replay	.688$_{\pm.004}$	.000$_{\pm.000}$	.938$_{\pm.007}$	.750$_{\pm.014}$	.000$_{\pm.000}$	92.20$_{\pm.25}$

5 Conclusion

In this work, we propose a novel untraining setting called Data-Free Untraining in TTA, which addresses the task of removing problematic sample influence while preserving useful adaptation and source knowledge. For this setting, we propose a batch-level untraining method based on a parameter-efficient TTA structure. Experiments on two corruption benchmarks show that our method performs well under different untraining requests.

AI USE STATEMENT

In this work, we used generative AI tools to assist with code implementation, language polishing and visual refinement. The authors manually reviewed all AI-assisted code, text, and layout. We take responsibility for the final content of this work, including text and artifacts produced with the aid of generative AI.

REFERENCES

Sk Miraj Ahmed, Umit Yigit Basaran, Dripta S. Raychaudhuri, Arindam Dutta, Rohit Kundu, Fahim Faisal Niloy, Basak Guler, and Amit K. Roy-Chowdhury. Towards source-free machine unlearning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4948–4957, 2025.

Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 141–159, 2021.

Yaohui Cai, Zhewei Yao, Zhen Dong, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. ZeroQ: A novel zero shot quantization framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13169–13178, 2020.

Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 463–480, 2015.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 39–57, 2017.

Hanting Chen, Yunhe Wang, Chang Xu, Zhaohui Yang, Chuanjian Liu, Boxin Shi, Chunjing Xu, Chao Xu, and Qi Tian. Data-free learning of student networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3514–3522, 2019.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

Arnav Devalapally, Poornima Jain, Kartik Srinivas, and Vineeth N. Balasubramanian. $\oslash$ source models leak what they shouldn't $\nrightarrow$: Unlearning zero-shot transfer in domain adaptation through adversarial optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1543–1553, 2026.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

European Parliament and Council. Regulation (EU) 2016/679 (general data protection regulation). *Official Journal of the European Union*, L 119:1–88, 2016. URL `https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679`. Article 17.

Chongyu Fan, Jiancheng Liu, Yihua Zhang, Eric Wong, Dennis Wei, and Sijia Liu. SalUn: Empowering machine unlearning via gradient-based weight saliency in both image classification and generation. In *International Conference on Learning Representations*, 2024.

Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In *International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 3762–3773. PMLR, 2020.

Jack Foster, Stefan Schoepf, and Alexandra Brintrup. Fast machine unlearning without retraining through selective synaptic dampening. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 12043–12051, 2024.

Shashwat Goel, Ameya Prabhu, Philip Torr, Ponnurangam Kumaraguru, and Amartya Sanyal. Corrective machine unlearning. *Transactions on Machine Learning Research*, 2024.

Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9304–9312, 2020.

Taesik Gong, Jongheon Jeong, Taewon Kim, Yewon Kim, Jinwoo Shin, and Sung-Ju Lee. NOTE: Robust continual test-time adaptation against temporal correlation. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27253–27266, 2022.

Laura Graves, Vineel Nagisetty, and Vijay Ganesh. Amnesiac machine learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 11516–11524, 2021.

Shurui Gui, Xiner Li, and Shuiwang Ji. Active test-time adaptation: Theoretical analyses and an algorithm. In *International Conference on Learning Representations*, 2024.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1321–1330. PMLR, 2017.

Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens van der Maaten. Certified data removal from machine learning models. In *International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3832–3842. PMLR, 2020.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019.

HHS. Summary of the HIPAA privacy rule. U.S. Department of Health and Human Services, 2025. URL `https://www.hhs.gov/hipaa/for-professionals/privacy/laws-regulations/index.html`. Content last reviewed March 14, 2025. Accessed September 26, 2026.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 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.

Frederik Kunstner, Lukas Balles, and Philipp Hennig. Limitations of the empirical Fisher approximation for natural gradient descent. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Meghdad Kurmanji, Peter Triantafillou, Jamie Hayes, and Eleni Triantafillou. Towards unbounded machine unlearning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 1957–1987, 2023.

Jonghyun Lee, Dahuin Jung, Saehyung Lee, Junsung Park, Juhyeon Shin, Uiwon Hwang, and Sungroh Yoon. Entropy is not enough for test-time adaptation: From the perspective of disentangled factors. In *International Conference on Learning Representations*, 2024.

Puwei Lian, Xiao Ke, Zhou Tan, Jianping Cai, and Ximeng Liu. Achieving zero-glance unlearning with data-free inversion and selective parameters suppression. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, pp. 1–9, 2025.

Jiaming Liu, Senqiao Yang, Peidong Jia, Renrui Zhang, Ming Lu, Yandong Guo, Wei Xue, and Shanghang Zhang. ViDA: Homeostatic visual domain adapter for continual test time adaptation. In *International Conference on Learning Representations*, 2024.

Sijia Liu, Yuanshun Yao, Jinghan Jia, Stephen Casper, Nathalie Baracaldo, Peter Hase, Yuguang Yao, Chris Yuhao Liu, Xiaojun Xu, Hang Li, Kush R. Varshney, Mohit Bansal, Sanmi Koyejo, and Yang Liu. Rethinking machine unlearning for large language models. *Nature Machine Intelligence*, 7 (2):181–194, 2025.

James Martens. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21(146):1–76, 2020.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In *International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 16888–16905. PMLR, 2022.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Zhiquan Wen, Yaofo Chen, Peilin Zhao, and Mingkui Tan. Towards stable test-time adaptation in dynamic wild world. In *International Conference on Learning Representations*, 2023.

Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. In *International Conference on Learning Representations*, 2021.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 3–18, 2017.

Xian Sun, Peijin Wang, Zhiyuan Yan, Feng Xu, Ruiping Wang, Wenhui Diao, Jin Chen, Jihao Li, Yingchao Feng, Tao Xu, Martin Weinmann, Stefan Hinz, Cheng Wang, and Kun Fu. FAIR1M: A benchmark dataset for fine-grained object recognition in high-resolution remote sensing imagery. *ISPRS Journal of Photogrammetry and Remote Sensing*, 184:116–130, 2022.

Eleni Triantafillou, Ahmed Imtiaz Humayun, Mónica Ribero, Alexander Matt Turner, Michael C. Mozer, and Georgios Kaissis. Is your algorithm unlearning or untraining? *arXiv preprint arXiv:2604.07962*, 2026.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021.

Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7201–7211, 2022.

Yinjun Wu, Edgar Dobriban, and Susan Davidson. DeltaGrad: Rapid retraining of machine learning models. In *International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 10355–10366. PMLR, 2020.

Hongxu Yin, Pavlo Molchanov, Jose M. Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K. Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via DeepInversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8715–8724, 2020.

Longhui Yuan, Binhui Xie, and Shuang Li. Robust test-time adaptation in dynamic scenarios. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15922–15932, 2023.

Marvin Zhang, Sergey Levine, and Chelsea Finn. MEMO: Test time robustness via adaptation and augmentation. In *Advances in Neural Information Processing Systems*, volume 35, pp. 38629–38642, 2022.

Bowen Zhao, Chen Chen, and Shu-Tao Xia. DELTA: Degradation-free fully test-time adaptation. In *International Conference on Learning Representations*, 2023.

Yu Zhou, Dian Zheng, Qijie Mo, Renjie Lu, Kun-Yu Lin, and Wei-Shi Zheng. Decoupled distillation to erase: A general unlearning method for any class-centric tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20350–20359, 2025.

A SUPPLEMENTARY DETAILS

Figure 1 illustrates the proposed setting: some incoming batches in a continual TTA stream contain problematic samples. Untraining targets the affected historical batches while preserving useful adaptation from the remaining stream. The following subsections provide further details of Fisher-based parameter importance and gradient projection.

A.1 FISHER-BASED PARAMETER IMPORTANCE

Fisher information and parameter importance. For fixed backbone weights θ_0, let $s = \nabla_\psi \log p_\psi(y \mid x)$. The Fisher information matrix is $\mathcal{I}(\psi) = \mathbb{E}_x \mathbb{E}_{y\sim p_\psi(\cdot|x)}[ss^\top]$. For a differentiable predictive distribution with finite score moments, its diagonal measures local sensitivity to individual parameters (Martens, 2020). This motivates diagonal importance weights in EWC (Kirkpatrick et al., 2017). In our bank, $F_{t,c,j}$ stores an importance score for parameter ψ_j in the vector indexed by class c. Figure A.1 illustrates this indexing.

Class-wise update. Following Eq. (5), we group reliable samples used by the base TTA update according to their predicted classes. For each nonempty class group, $\ell_{t,c}$ denotes its mean adaptation loss. The gradient for parameter ψ_j and the corresponding importance update are

$$g_{t,c,j} = \frac{\partial \ell_{t,c}}{\partial \psi_{t,j}}, \qquad F_{t,c,j} = \beta_F F_{t-1,c,j} + (1-\beta_F) g_{t,c,j}^2, \quad 0 \le \beta_F < 1. \tag{A.1}$$

Vectors are initialized at zero. Classes with no reliable samples retain their previous vectors. Each class therefore accumulates importance scores from its own adaptation loss without storing historical raw inputs.

Interpretation. At a differentiable point, holding the selected class group fixed, a small change to parameter ψ_j alone gives $\Delta\ell_{t,c} = g_{t,c,j}\Delta\psi_j + o(|\Delta\psi_j|)$. Thus, $g_{t,c,j}^2$ measures local sensitivity of the class-c adaptation loss, and the moving average tracks this sensitivity over time. The score is a squared gradient of a class-mean loss, distinct from an average of squared per-sample likelihood gradients in the empirical Fisher (Kunstner et al., 2019). In Eq. (10), larger retained-class weights penalize changes to the corresponding parameters more strongly.

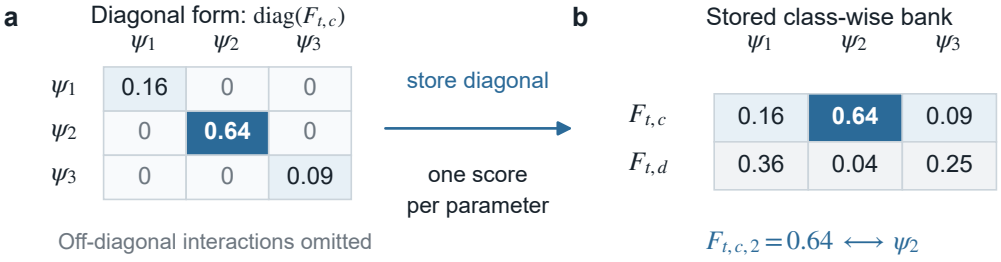


Figure A.1: **Fisher-based parameter importance** (illustrative values). (a) Diagonal matrix for class c; (b) stored vectors for classes c and d. The highlighted value $F_{t,c,2} = 0.64$ weights parameter ψ_2 for class c. Off-diagonal interactions are omitted.

A.2 ANALYSIS OF GRADIENT PROJECTION

The projection in Eq. (12) defines a constrained feature update that follows the untraining objective while preserving the current token's linear class score. For $(\hat{x}, c)$, let $h = h_\psi(\hat{x})$, let $\ell(h)$ be its contribution to $\mathcal{L}_{\text{forget}}$, and set $G = \nabla_h \ell(h)$. Assume that ℓ is differentiable at h, with $h \neq 0$ and $\|W_c\|_2^2 \ge \epsilon > 0$, so the projection denominator is not clipped.

Constrained feature update. For the frozen linear classifier, a direct feature change Δh changes the class-c score by $W_c^\top \Delta h$. To preserve this score while following the local untraining objective, consider, for $\eta > 0$,

$$\min_{\Delta h} \; G^\top \Delta h + \tfrac{1}{2\eta}\|\Delta h\|_2^2 \quad \text{subject to} \quad W_c^\top \Delta h = 0. \tag{A.2}$$

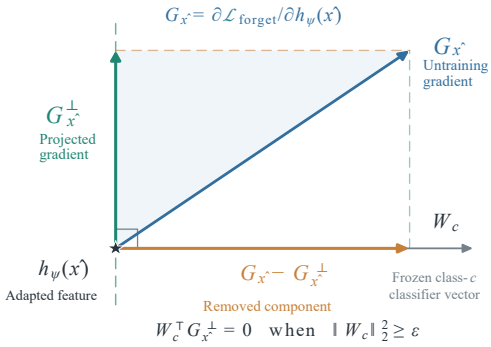


Figure A.2: Feature-gradient decomposition at $h_\psi(\hat{x})$ (star). Blue: untraining gradient $G_{\hat{x}}$; orange: removed component along W_c; green: projected component $G_{\hat{x}}^{\perp}$ in Eq. (12).

Completing the square shows that this selects the feasible update closest to the unconstrained step $-\eta G$. The objective is strictly convex, and Lagrangian stationarity gives $G + \Delta h/\eta + \nu W_c = 0$. Using the constraint yields $\nu = -W_c^\top G / \|W_c\|_2^2$, hence the unique solution is

$$\Delta h^\star = -\eta P_c G = -\eta G^\perp, \qquad P_c = I - \frac{W_c W_c^\top}{\|W_c\|_2^2}. \tag{A.3}$$

Thus, the projection in Eq. (12) is the smallest Euclidean modification of the original feature gradient needed to satisfy the class-score constraint.

Descent in feature space. Since $P_c^\top = P_c$ and $P_c^2 = P_c$, the constrained step satisfies

$$G^\top \Delta h^\star = -\eta \|P_c G\|_2^2, \qquad \ell(h + \Delta h^\star) = \ell(h) - \eta \|P_c G\|_2^2 + o(\eta). \tag{A.4}$$

When $P_c G \neq 0$, a sufficiently small direct feature step therefore decreases the untraining loss while preserving the class-c linear score. For the cosine objective, the feature norm can change while the linear score stays fixed, allowing the normalized alignment to decrease. When $P_c G = 0$, the constrained first-order update is zero. These properties motivate retaining the component that reduces the objective without directly suppressing the feature along the class direction.

Backpropagation to parameters. For a parameter update, the first-order feature change also depends on the network Jacobian. Let $J = \partial h_\psi(\hat{x})/\partial \psi$. Backpropagating only the projected forget gradient and taking an ordinary gradient step instead gives

$$\Delta \psi = -\eta J^\top P_c G, \qquad \Delta h = -\eta J J^\top P_c G + o(\eta). \tag{A.5}$$

The resulting first-order class-score change is $-\eta W_c^\top J J^\top P_c G$, which need not vanish even though $W_c^\top P_c G = 0$. The implemented update further combines multiple tokens, Adam preconditioning, and the retention losses. The exact score-preservation and descent results above therefore apply to the direct feature step; retention after parameter optimization is evaluated experimentally. For $0 < \|W_c\|_2^2 < \epsilon$, denominator clipping gives the residual $W_c^\top G^\perp = (W_c^\top G)(1 - \|W_c\|_2^2/\epsilon)$, so exact orthogonality requires the stated norm condition.

B Algorithm Details

B.1 Untraining Algorithm

Algorithm B.1 details the online procedure, including history updates, token reconstruction, correction, and subsequent TTA.

A nonconsecutive request is processed as a chronological sequence of contiguous intervals. We sort and deduplicate the requested indices, partition them into maximal contiguous intervals, and carry the corrected state from one interval to the next without replaying intervening batches between interval corrections.

Algorithm B.1 Parameter-efficient data-free untraining in TTA

Require: Frozen weights θ_0, stream $\{X_t\}_{t=1}^{T}$, untraining requests, TTA rule $\mathcal{A}$, budgets (R, L, H), and M untraining steps
Ensure: Adapted model $\Theta_T = (\theta_0, \psi_T)$
Initialize ψ_0, model-state history, and the class-wise Fisher bank
for $t = 1, \ldots, T$ **do**
  Set $\psi \leftarrow \psi_{t-1}$; run TTA on X_t until completion or request arrival
  if request $q = (I, t_{\mathrm{req}})$ is received **then**
    Set $\psi_{\mathrm{pre}} \leftarrow \psi$; select the latest eligible ψ_{ref} before $\min I$, or ψ_0 if unavailable
    Retrieve distinct records with $\mathcal{B}_s \cap I \neq \varnothing$; set $Y_I \leftarrow \bigcup_s \mathcal{Y}_s$ over these records
    for all retrieved record–class pairs (s, c) with $c \in \mathcal{Y}_s$ **do**
      Reconstruct $\hat{x}_{s,c}$ from noise by Eq. (6), with ψ_s and ψ_{ref} fixed
    end for
    Form $\widehat{\mathcal{D}}_u$ by Eq. (7) and Fisher weights $\widehat{F}$ by Eqs. (9)–(10)
    If $\lambda_p > 0$, reconstruct $\widehat{\mathcal{D}}_r$ for classes in Y_I by Eq. (14)
    for $i = 1, \ldots, M$ **do**
      Project the $\mathcal{L}_{\mathrm{forget}}$ feature gradient by Eq. (12); backpropagate to obtain $g^{\perp}_{\mathrm{forget},i}$
      Form g_i by Eq. (13); add $\lambda_p \nabla_\psi \mathcal{L}_{\mathrm{retain}}$ if $\lambda_p > 0$ and $\widehat{\mathcal{D}}_r \neq \varnothing$
      Update $\psi \leftarrow \psi - \eta_u g_i$
    end for
    Set $\psi_u \leftarrow \psi$; resume TTA as $\Theta_t \leftarrow \mathcal{A}((\theta_0, \psi_u), X_t)$
    Invalidate archived records whose blocks overlap I
  else
    Set $\Theta_t \leftarrow (\theta_0, \psi)$
  end if
  Extract ψ_t from Θ_t; predict X_t to obtain Y_t
  Update the Fisher bank by Eq. (5) and store the exact record $(\{t\}, \psi_t, Y_t)$
  Archive blocks of L older batches by Eq. (4); enforce the record limits R and H
end for
return Θ_T

Algorithm B.2 uses CORRECTINTERVAL for the reconstruction, correction, and history maintenance of Algorithm B.1. All intervals must satisfy the history coverage and invalidation rules in Sec. 3.2. Within one request, interval corrections run back to back; between requests, TTA resumes. The experiments evaluate contiguous requests and successive request sequences.

Algorithm B.2 Interval-wise processing of a nonconsecutive untraining request

Require: Current model and optimizer state S, available history $\mathcal{H}$, Fisher bank $\mathcal{F}$, requested batch-index set I
Ensure: Corrected state and updated records, or a request outside the available history
Sort and deduplicate I; partition it into maximal contiguous intervals $I_1, \ldots, I_K$
Order intervals by their earliest batch index
Check that each requested interval is covered by valid history records
if the plan is unsupported **then**
  return Unsupported request
end if
for $k = 1, \ldots, K$ **do**
  Set the pre-correction state to the current S
  $(S, \mathcal{H}, \mathcal{F}) \leftarrow$ CORRECTINTERVAL$(S, \mathcal{H}, \mathcal{F}, I_k)$
  Use the updated state for the next interval
end for
return $(S, \mathcal{H}, \mathcal{F})$; resume the live stream once the request completes

For requests arriving over time, each method maintains its own evolving state. It resumes TTA after correction and receives the next request at the scheduled arrival point. Replay is recomputed using the cumulative union of requested batches, while the stream order and evaluation images are matched across methods. Algorithm B.3 describes this comparison protocol.

Model updates between requests. After ViDA correction, the merged adapter parameters are copied to both the EMA teacher and the online model. Optimizer moments for the replaced online parameters are cleared; the correction optimizer is local to the request. TENT continues with its corrected LayerNorm parameters. The class-wise Fisher bank and running feature statistics remain online summaries and continue to be updated during subsequent TTA. Historical records follow the

Algorithm B.3 Evaluation of successive untraining requests with continued TTA

Require: Ordered requests $(I_i, \tau_i)_{i=1}^{Q}$, matched stream and seed, methods $\mathcal{M}$, evaluation sets $\mathcal{E}_i$, initial states
Ensure: Evaluation results for each method and request
Initialize an independent model and optimizer state S_m for each method m; set $U \leftarrow \varnothing$
for $i = 1, \ldots, Q$ **do**
  Adapt each method on the stream until request arrival τ_i
  $U \leftarrow U \cup I_i$
  Obtain the matched Replay state by excluding U from the historical stream
  for all methods $m \in \mathcal{M}$ **do**
    Save S_m and its runtime metadata before correction
    Apply method m to I_i; use Algorithm B.2 when interval scheduling is required
    Evaluate the corrected state against matched Replay on $\mathcal{E}_i$
    Save the corrected state and resume TTA from that state
  end for
end for

coverage and invalidation rules in Sec. 3.2 and Algorithm B.1. The input iterator continues from the request boundary, preserving the remaining stream order. Replay is constructed separately from the source state with the cumulative requested batches excluded.

B.2 Sequential Adaptation and Retention

After each request, TTA uses predictions from the corrected model to compute subsequent updates. Table C.1 separates retained accuracy from Replay agreement: on ImageNet-C, same-class accuracy matches Replay at the later requests, while the confidence gap increases. Repeated correction can therefore affect class accuracy and confidence agreement differently.

We evaluate reconstructed inputs and the retain loss in separate ablations. Table 7 holds the correction procedure and reconstructed retain inputs fixed, and compares reconstructed untraining tokens with real images. Table 6 isolates the retain loss in the sensitive-data setting. The synthetic-data experiments use $\lambda_p = 0$ throughout their request sequences.

C Additional Results

C.1 Sequential Untraining

Three requests on two datasets. We test how repeated requests affect retained knowledge across two adaptation streams. Table C.1 compares Ours with cumulative Retain-Only Replay on ImageNet-C and CIFAR-100-C under the protocol in Sec. 4.2, including attack-score gaps and the composite Score.

Table C.1: **Three successive sensitive-data requests with ViDA.** Oracle is cumulative Retain-Only Replay; results are measured immediately after each correction.

ImageNet-C							
Model	Request	F-Conf	CF↓	SC↑	R↑	MIA↓	Score↑
Ours	First	$.691_{\pm.007}$	$.003_{\pm.003}$	$.875_{\pm.010}$	$.750_{\pm.005}$	$.028_{\pm.013}$	$89.85_{\pm.38}$
	Second	$.589_{\pm.010}$	$.107_{\pm.008}$	$.812_{\pm.006}$	$.625_{\pm.005}$	$.338_{\pm.005}$	$74.80_{\pm.44}$
	Third	$.684_{\pm.005}$	$.122_{\pm.009}$	$.812_{\pm.007}$	$.812_{\pm.014}$	$.151_{\pm.008}$	$83.78_{\pm.39}$
Oracle	First	$.688_{\pm.004}$	$.000_{\pm.000}$	$.938_{\pm.007}$	$.750_{\pm.014}$	$.000_{\pm.000}$	$92.20_{\pm.25}$
	Second	$.482_{\pm.002}$	$.000_{\pm.000}$	$.812_{\pm.012}$	$.625_{\pm.012}$	$.000_{\pm.000}$	$85.93_{\pm.19}$
	Third	$.562_{\pm.013}$	$.000_{\pm.000}$	$.812_{\pm.014}$	$.875_{\pm.008}$	$.000_{\pm.000}$	$92.18_{\pm.28}$
CIFAR-100-C							
Model	Request	F-Conf	CF↓	SC↑	R↑	MIA↓	Score↑
Ours	First	$.766_{\pm.004}$	$.053_{\pm.015}$	$.782_{\pm.014}$	$.781_{\pm.004}$	$.108_{\pm.010}$	$85.05_{\pm.47}$
	Second	$.657_{\pm.011}$	$.011_{\pm.005}$	$.531_{\pm.006}$	$.742_{\pm.011}$	$.043_{\pm.001}$	$80.48_{\pm.29}$
	Third	$.688_{\pm.007}$	$.100_{\pm.008}$	$.743_{\pm.008}$	$.711_{\pm.002}$	$.005_{\pm.005}$	$83.73_{\pm.25}$
Oracle	First	$.819_{\pm.013}$	$.000_{\pm.000}$	$.828_{\pm.005}$	$.781_{\pm.010}$	$.000_{\pm.000}$	$90.23_{\pm.37}$
	Second	$.668_{\pm.006}$	$.000_{\pm.000}$	$.609_{\pm.013}$	$.773_{\pm.001}$	$.000_{\pm.000}$	$84.55_{\pm.35}$
	Third	$.788_{\pm.007}$	$.000_{\pm.000}$	$.828_{\pm.014}$	$.727_{\pm.003}$	$.000_{\pm.000}$	$88.88_{\pm.42}$

On ImageNet-C, same-class accuracy matches Replay at the second and third requests, while CF-Gap increases from 0.003 to 0.122. On CIFAR-100-C, same-class accuracy falls from 0.782 to 0.531 at

the second request and recovers to 0.743 at the third; retained-stream accuracy stays close to Replay throughout. CF-Gap is 0.100 at the third request. The five-request corrupted-data experiment with ViDA is reported in Table 5 and Sec. 4.2. The two streams show different confidence-gap trends while retaining useful classification accuracy across requests.

Synthetic corruption examples. Figure C.1 illustrates the localized-glare corruption used for the repeated corrupted-data requests in Table 5. The race-car, trailer-truck, and fire-engine classes are used in this experiment. Each pair shows the same image before and after applying the pixel-space corruption operator in Eq. (D.1). The examples use a gain of 0.90 and radius of 0.80 to make the contamination visually apparent.

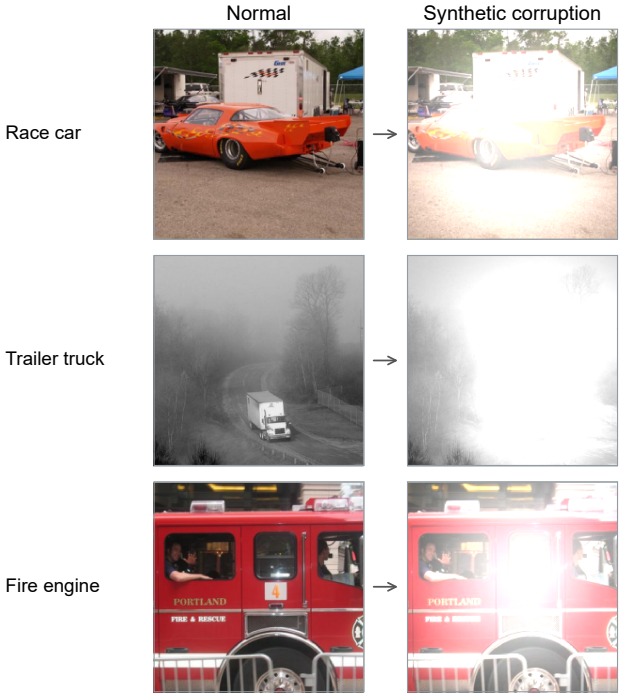


Figure C.1: **Examples of synthetic corruption.** Normal inputs (left) and their synthetically corrupted counterparts (right). Each row shows the same image before and after adding localized glare for one class used in the sequential corrupted-data experiment.

C.2 Prediction Agreement with Replay

Evaluation protocol. We compare individual predictions and probability distributions from the first-request ImageNet-C checkpoints. All methods use the same fixed evaluation pools: 23 FAIR1M target images, 1,000 normal ImageNet-C images, 250 same-class source images, and 1,000 other-class source images. These pools measure predictive agreement with Replay. Checkpoint and sampling details are in Appendix D.4.

Let p_i and o_i be the evaluated model's and Replay's probability vectors for the same image. We fix c_i to the No-Untrain model's predicted class for every method and compute

$$A_{\text{pred}} = N^{-1} \sum_i \mathbf{1}[\arg\max p_i = \arg\max o_i], \tag{C.1}$$

$$E_{\text{conf}} = N^{-1} \sum_i |p_{i,c_i} - o_{i,c_i}|, \tag{C.2}$$

$$d_i = \tfrac{1}{2}\text{KL}(p_i \| m_i) + \tfrac{1}{2}\text{KL}(o_i \| m_i), \quad m_i = (p_i + o_i)/2. \tag{C.3}$$

Confidence MAE takes the absolute difference before averaging, preventing positive and negative errors from canceling. We report the mean and 95th percentile of the per-image JS divergences d_i, using natural logarithms. Table C.2 reports prediction agreement, confidence MAE, and the mean and P95 JS divergence.

Table C.2: Sample-level Replay agreement. Agreement is a percentage; JS divergence and confidence MAE are unscaled.

Method	Agreement (%)↑	JS mean ↓	JS P95 ↓	Conf. MAE ↓
FAIR1M target images ($N = 23$)				
Ours	91.30	0.0721	0.1449	0.1016
No-Untrain	95.65	0.1214	0.3626	0.2751
Replay	100.00	0.0000	0.0000	0.0000
Normal images ($N = 1,000$)				
Ours	91.70	0.0137	0.0611	0.0378
No-Untrain	92.70	0.0090	0.0433	0.0394
Replay	100.00	0.0000	0.0000	0.0000
Same-class source images ($N = 250$)				
Ours	96.00	0.0069	0.0200	0.0109
No-Untrain	98.40	0.0029	0.0187	0.0184
Replay	100.00	0.0000	0.0000	0.0000
Other-class source images ($N = 1,000$)				
Ours	91.40	0.0149	0.0655	0.0471
No-Untrain	92.10	0.0091	0.0471	0.0409
Replay	100.00	0.0000	0.0000	0.0000

Agreement with Replay. On FAIR1M target images, our method reduces mean JS divergence from 0.1214 to 0.0721, JS P95 from 0.3626 to 0.1449, and confidence MAE from 0.2751 to 0.1016 relative to No-Untrain. Top-1 agreement is 91.30% (21/23), compared with 95.65% (22/23) for No-Untrain. Correction thus brings the probability distributions closer to Replay with one fewer top-1 match.

On normal, same-class, and other-class images, our agreement rates are 91.70%, 96.00%, and 91.40%, respectively. No-Untrain has higher top-1 agreement and lower mean and P95 JS divergence in all three groups. Confidence MAE decreases from 0.0394 to 0.0378 on normal images and from 0.0184 to 0.0109 on same-class images, but increases from 0.0409 to 0.0471 on other-class images. The correction reduces distribution differences on target images, with over 91% prediction agreement on each retained-image set.

C.3 Runtime and Memory Usage

We profile our method with ViDA on the first ImageNet-C sensitive-data request and five successive corrupted-data requests, using four Tesla P100-SXM2 GPUs (16 GB each), PyTorch 1.10.0, and CUDA 11.3. The sensitive-data run uses 40 untraining-token inversion steps, 30 retain-token reconstruction steps, and 10 correction steps, with learning rate 10^{-4}, $\lambda_f = 4$, $\lambda_p = 2$, and $\lambda_e = 5 \times 10^5$. The corrupted-data sequence uses 50 inversion steps and 10 correction steps per request, with retain-token reconstruction disabled ($\lambda_p = 0$). Table C.3 reports each request, taking the maximum across ranks.

We measure correction time up to the returned prediction, synchronizing CUDA at the start and end of each stage. Active runtime excludes diagnostic evaluation and cache I/O. Model initialization, preceding TTA, and subsequent checkpoint saving are measured outside this interval. Correction time includes construction of the correction model, Fisher aggregation, optimization, and parameter merging. The total for each GPU process includes history retrieval, sample selection, and preparation. We report the maximum total across the four processes.

Table C.3: Runtime and memory usage of our method on ImageNet-C. GPU: peak allocated memory; State: persistent history and Fisher storage. Entries are maxima across four ranks.

Setting	Req.	Invert (s)	Retain (s)	Correct (s)	Active (s)	GPU (GiB)	State (GiB)
Privacy	1	25.43	23.69	33.57	83.04	1.43	13.21
Pollution	1	27.42	0.00	22.88	56.88	1.43	12.11
	2	27.22	0.00	26.93	58.92	1.43	13.41
	3	26.68	0.00	22.09	55.05	1.43	15.33
	4	26.81	0.00	22.57	57.37	1.43	17.71
	5	26.85	0.00	22.22	56.33	1.43	18.80

For the sensitive-data request, untraining and retain reconstruction together account for 58–61% of each rank's active time. Across the five corrupted-data requests, active time remains between 55.05 and 58.92 s (mean 56.91 s), and peak allocated GPU memory stays near 1.43 GiB. Inversion takes 26.68–27.42 s per request, while correction takes 22.09–26.93 s. Including internal diagnostics,

correction-call wall time ranges from 87.32 to 90.90 s for the corrupted-data sequence. Runtime and peak GPU allocation remain similar across the five requests.

Persistent storage is measured separately by counting unique tensor storage at request entry and exit. It includes historical model states, their feature statistics, running archive accumulators, and the class-wise Fisher bank; Python container overhead and evaluation buffers are excluded. In the corrupted-data sequence, each rank keeps 50 exact records and 10 archived records. Persistent storage grows from 12.11 to 18.80 GiB as the stream progresses. On the rank with the largest state at each request, Fisher storage rises from 9.06 to 15.88 GiB, whereas history-related storage stays within 2.76–3.38 GiB. The Fisher bank therefore accounts for most of the storage growth, even though the number of historical records remains fixed.

D Implementation Details

We describe data preparation, baselines, hyperparameters, and evaluation.

D.1 Datasets and Data Preparation

Normal target streams. Normal samples are drawn from ImageNet-C or CIFAR-100-C outside the constructed target batches. ImageNet-C uses snow, fog, frost, brightness, and contrast at severity 1; CIFAR-100-C uses all 15 corruptions at severity 4. Normal samples retain these benchmark corruptions. The ImageNet-C loader resizes and center-crops images to the model input size, then normalizes RGB channels with means $(0.485, 0.456, 0.406)$ and standard deviations $(0.229, 0.224, 0.225)$. CIFAR-100-C contains 10,000 images per corruption and severity, derived from the same underlying test set at native resolution 32×32. The loader selects the severity slice and bilinearly upsamples it to 384×384 before normalization.

Sensitive inputs from FAIR1M. We use images from the ARJ21 and Boeing747 aircraft categories in FAIR1M as the sensitive component of mixed ImageNet-C target batches. These examples assess removal of requested-batch influence while retaining recognition of the broader aircraft class. The sample-level Replay comparison uses ARJ21 images.

Synthetic corruptions. Synthetic inputs are generated by applying pixel-space perturbations to existing dataset images. Supported perturbations are additive Gaussian noise, sinusoidal colored stripes, and localized glare. Pixel values are clipped to $[0, 1]$ and then normalized for model evaluation. For glare, normalized spatial coordinates $u, v \in [-1, 1]$ receive the field

$$I'(u, v) = \mathrm{clip}_{[0,1]} \left[I(u, v) + a \exp\left(-\frac{(u-u_0)^2 + (v-v_0)^2}{2r^2} \right) \right], \tag{D.1}$$

where the center is sampled uniformly within a configured central range. The implementation caps the effective gain at 0.90 and bounds the radius below by 0.05. The field is added to all RGB channels. Stripe orientation, phase, and color can be conditioned on the configured class.

For CIFAR-100-C, source images are selected by class and perturbed; teacher filtering retains variants with the intended predicted class and within the configured confidence range. Synthetic ImageNet variants are likewise filtered by a fixed teacher. Normal stream images retain their benchmark corruptions, whereas constructed inputs receive the additional perturbation.

D.2 Baseline Implementation

Experimental setup. Following Devalapally et al. (2026), we preserve each baseline's correction objective and replace unavailable historical inputs with signals from the target stream. For the main comparison, Ours and the five MU baselines use the same pretrained ViT-B/16 backbone, ViDA parameter subspace, and pre-correction model state. The corrupted stream, target batches, request intervals, and seeds are fixed across methods. Before TTA resumes, all models are evaluated on the same requested, same-class, and retained-stream images, with seed-matched Replay and identical metrics. Replay is the evaluation oracle; baseline objectives and optimization settings follow below.

Input data. The five adapted MU methods initialize from the pre-correction EMA teacher and update only ViDA parameters. They use class-conditioned forget proxies and the current unlabeled TTA batch; past input images are not reloaded. Class-conditioned baselines receive the benchmark's configured target classes for proxy construction; our method retrieves predicted classes from the records indexed by the request.

Retain samples from the current batch. All five MU baselines select real retain inputs from the batch available when the request arrives. The pre-correction teacher supplies confidence scores, soft targets, and pseudo-labels. Selection favors confident predictions, with a highest-confidence fallback when too few images pass the threshold. SCADA-UL-style, DELETE-PEFT, SalUn-PEFT, and SCRUB-PEFT use them for teacher–student retention losses; SSD-PEFT uses them to estimate retain-side parameter importance for selective dampening. Our intra-class retain loss uses reconstructed tokens.

Proxy generation. The proxy initializer uses patch tokens from current TTA images plus Gaussian noise. Let Z_0 denote the resulting initialization and c the configured forget class. With the teacher parameters fixed and its logits denoted by $z(Z)$, we minimize

$$\mathcal{L}_{\text{proxy}} = \text{CE}(z(Z), c) + w_m \max\{0, \max_{j \neq c} z_j(Z) - z_c(Z) + m\} + w_2 \,\text{mean}[(Z - Z_0)^2]. \quad \text{(D.2)}$$

Optimization uses Adam at learning rate 0.05. The shared defaults are $w_m = 1$, $w_2 = 10^{-4}$, and initialization noise standard deviation 0.05.

Baseline objectives. **SCADA-UL-style** zeros the target-class teacher probability, renormalizes the remaining classes, and matches this distribution on forget proxies, alongside retain distillation and pseudo-label cross-entropy. **DELETE-PEFT** adds dark-knowledge distillation over non-target logits and current-batch retain distillation. **SSD-PEFT** estimates squared-gradient importance on forget proxies and pseudo-labeled retain images. It selects parameters with high forget-to-retain importance ratios and dampens them in proportion to retain importance, without subsequent retain fine-tuning. **SalUn-PEFT** obtains a saliency mask from non-target random-label gradients and applies it to the combined forget and retain updates. **SCRUB-PEFT** maximizes teacher–student KL on forget proxies and minimizes retain KL, with a random-label term to break the zero-gradient initialization. The latter two methods use retain distillation without retain cross-entropy.

Optimization and model updates. Gradient-based corrections use a request-local Adam optimizer. Method-specific update budgets and loss coefficients are fixed to each setting's first-request configuration. After correction, adapter parameters are merged into the EMA model and copied to the online model; optimizer moments for replaced online parameters are cleared. SSD applies its importance-based dampening directly.

D.3 Hyperparameters and Request Schedules

The backbones have 1,000 and 100 output classes for ImageNet-C and CIFAR-100-C, respectively. The implementation uses `vit_base_patch16_224` with an ImageNet checkpoint and `vit_base_patch16_384` with a CIFAR-100 checkpoint. ImageNet-C inputs are 224×224, and CIFAR-100-C inputs are upsampled to 384×384. ViDA updates compact adapters with an exponential moving average (EMA) teacher. TENT updates LayerNorm affine parameters in a single model.

Tokens follow the backbone's positional embedding: the two ViT-B/16 models use 197 and 577 positions, respectively, including the class token, with dimension 768. Real images are converted through patch embedding, class-token concatenation, and positional embedding before entering the transformer blocks. Inversion directly optimizes tokens at this feature interface.

The ViDA implementation reconstructs tokens with Adam (Kingma & Ba, 2015) at learning rate 0.05, using a default margin $\kappa = 15$ and norm weight $\beta_{\text{norm}} = 10^{-4}$. Correction uses Adam with default learning rate $\eta_u = 10^{-4}$ and $M = 10$ steps. The main-comparison scripts specify 40 token-reconstruction steps; the five-request synthetic-corruption script specifies 50. The main sensitive-data protocol enables the retain loss with $\lambda_p = 0.25\lambda_f$, whereas the synthetic-corruption protocol sets $\lambda_p = 0$. For each dataset and experimental setting, loss weights are fixed at their

first-request values throughout the sequence. Appendix C.3 gives the time and space measurements and profiling configuration.

Ablation configurations. The component ablations use the first-request ImageNet-C configuration of Ours in Table 2, with the same batch composition, pre-correction ViDA state, and update budget. The full method uses $\lambda_p = 0.25\lambda_f$; *w/o retain loss* sets $\lambda_p = 0$. The *w/o Fisher weights* variant sets $\widehat{F}_j = 1$ in Eq. (10), retaining the quadratic penalty and λ_e. The *w/o EWC* variant sets $\lambda_e = 0$ in Eq. (16), removing that penalty.

Request schedules and batch composition. After initial TTA, we insert several dozen consecutive target batches and request their removal. TTA resumes after correction, and later requests repeat this sequence on the continuing stream. The main comparison uses one request; sequential experiments use three or five. Evaluation immediately follows each correction. Nonconsecutive requests use the interval procedure in Appendix B.1. The global batch size is 64.

Sensitive-data batches contain 30 target and 34 normal samples on ImageNet-C, or 32 of each on CIFAR-100-C. Synthetic-corruption batches contain 64 perturbed samples and no normal samples. All samples in an indexed batch belong to the untraining request, including the normal component of a mixed batch. Exposure counts include repeated presentations of the same image.

D.4 Evaluation Protocol

The confidence and accuracy evaluation samples up to 128 requested, 128 retained, and 128 same-class retained images per request, subject to availability, with images fixed across model states. The membership-labeled MIA evaluation below uses a separate evaluation set with no shared source images.

The main protocol pairs seeds $0, 1, 2, 3, 4$ across methods. CF-Gap and accuracy are computed per seed before aggregation; sample standard deviations use denominator $n - 1$. MIA-Gap and the dependent Score are likewise computed against each seed's matched Replay checkpoints before aggregation. An attack split seed does not create a new model-training run.

Prediction agreement with Replay. The supplementary comparison uses first-request ViDA EMA checkpoints. Its target pool contains FAIR1M ARJ21 aircraft images; the other pools use the severity-1 ImageNet-C corruptions specified above. The 2,273 images are disjoint across pools by image hash and fixed across Ours, No-Untrain, and Replay. ViT-B/16 receives 224×224 inputs; all 1,000 class probabilities are saved. No-Untrain defines each confidence class, and JS P95 uses linear interpolation. Results are point estimates for one checkpoint set.

Entropy-based membership inference. For each request, members are the designated special-data images that actually participated in TTA. Nonmembers come from the same class and source but were never exposed to TTA. Privacy nonmembers are drawn from the corresponding raw fine-grained class after excluding all request images by decoded-pixel hash. Synthetic nonmembers use confirmed unseen source images, the original perturbation generator, and the same clean-teacher acceptance rule. The privacy evaluation compares filtered members with nonmembers sampled from the raw data. The attack evaluates the designated sensitive or corrupted samples within each requested batch.

We retain one image per source and at most 500 sources per class/source stratum, then split members and nonmembers separately into 50% calibration and 50% test sets (split seed 2026). Variants of one source cannot cross sets. For each stratum, normalized entropy $h(p) = -\sum_{c=1}^{C} p_c \log p_c / \log C$ is standardized using only the request-arrival calibration predictions. A logistic attack is fitted to true membership labels (member 1, nonmember 0), using balanced weights, 800 Adam steps, learning rate 0.05, and weight regularization 10^{-3}. The attack, normalization, and decision threshold 0.5 remain fixed across all corrected models and cumulative Replay; target models remain frozen.

Let F_{test} contain the held-out requested members and $a_{s(i)}$ be the attack for image i's class/source stratum. We compute

$$G_{\text{MIA}} = \left| \frac{1}{|F_{\text{test}}|} \sum_{i \in F_{\text{test}}} a_{s(i)}(h(p_u(x_i))) - \frac{1}{|F_{\text{test}}|} \sum_{i \in F_{\text{test}}} a_{s(i)}(h(p_o(x_i))) \right|. \tag{D.3}$$

Here p_u and p_o denote corrected-model and Replay predictions. Both attack probabilities and G_{MIA} lie in $[0, 1]$. We also compute ROC-AUC, balanced attack accuracy, TPR, and FPR on held-out members and nonmembers, averaging metrics equally across strata. A small gap measures agreement with Replay's attack scores; absolute attack performance is assessed separately. We report TPR at 1% or 0.1% FPR for strata with at least 100 or 1,000 test nonmembers, respectively. Saved manifests, predictions, attack parameters, and checkpoint hashes make the evaluation reproducible without rerunning TTA or untraining.

Evaluation across random seeds. Main-text metrics are aggregated across paired seeds; runtime and memory follow Appendix C.3.