

ORIGINAL RESEARCH

Open access

Efficient Machine Unlearning via Category-Aware Weight Modification and Continual Learning

Patrick O'Connor^{1*}, Grace Murphy¹, Niamh Walsh²

Abstract

Machine unlearning is vital for model optimization and privacy preservation, particularly when erasing entire data classes within classification frameworks. Nonetheless, current approaches frequently suffer from excessive computational overhead, such as calculating the inverse Hessian, or necessitate availability of the primary training set, which restricts their real-world utility. To address these limitations, we present Scrub-and-Learn, a category-aware weight adjusting framework engineered for the efficient removal of class-level information. Formulating unlearning as a sequential learning objective, our approach utilizes re-encoded labels from target class samples to steer parameter adjustments, successfully purging targeted information while maintaining the network's remaining capabilities. Empirical evaluations across several benchmarks show that our technique successfully wipes out designated categories—limiting the identification rate to under 5%—while maintaining the accuracy of the kept classes within a 4% variance compared to the baseline model.

Keywords Machine unlearning, Category-aware weight modification, Continual learning, Class-level forgetting

*Correspondence:

Patrick O'Connor

patrick.oconnor@ucc.ie

¹ Department of Machine Unlearning and Privacy-Preserving AI, School of Engineering, University College Cork, Cork, Ireland

² Department of Category-Aware Weight Modification, Faculty of Engineering, University of Galway, Galway, Ireland

Introduction

As machine learning technologies mature, the efficient erasure of sensitive information from models—while ensuring regulatory adherence and user privacy—has emerged as a vital area of study. Machine unlearning [1-5] has surfaced as a viable remedy, providing enhancements in privacy security, operational throughput, and model tuning. Through the selective erasure of target data information, unlearning wipes out sensitive details without necessitating a complete training restart, thereby defending user confidentiality [6-9], neutralizing adversarial backdoor injections [10, 11], and satisfying statutory mandates like the GDPR [12]. Furthermore, by purging distorted training data, unlearning refines model objectivity and interpretability [13, 14], showing its extensive utility.

The objective of machine unlearning is to eliminate the impact of target data from an established model, ensuring its behavior mirrors a state where it never encountered those samples, all without initiating a complete retraining loop [15]. Within classification objectives, the correlation between whether a model has interacted with a sample and its capacity to correctly assign its label is intricate. Because of the generalization capabilities inherent in models, unlearning techniques can preserve high classification accuracy even on unseen validation partitions. Moreover, investigations into core representative datasets [16-18] indicate that optimizing a network via a compact, highly representative data subset can still yield robust classification metrics. Consequently, within classification problems, the primary emphasis is directed toward erasing entire categories rather than isolated data points.

For classification settings, contemporary unlearning strategies generally follow two distinct trajectories. The initial trajectory concentrates on network parameters, seeking to purge the information regarding deleted records that resides within them. These strategies are frequently built upon influence functions [19] and incorporate paradigms like Certified Removal [20–22]. Such techniques typically necessitate the computation of the inverse Hessian matrix. Even though alternative approaches approximate this matrix via the Fisher Information Matrix (FIM) [23–25], the computational overhead remains substantial, restricting their scalability to extensive datasets and deep architectural networks.

The alternative trajectory treats the network as a baseline, formulates an appropriate objective function, and executes the unlearning process via sequential learning. This strategy bypasses the calculation of the Hessian matrix or the FIM, but requires an active dataset for ongoing optimization. For example, Variational Bayesian Unlearning [26], Saliency Unlearning [27], along with related strategies [28–32] depend on access to the primary training data. This constraint elevates data storage demands, as the training sets must be preserved indefinitely. Conversely, alternative approaches [33–37] only necessitate the data slated for removal, thereby lowering storage footprints. However, they rely on supplemental generative networks to synthesize artificial data to offset the absence of the remaining data.

To surmount the hurdles of intensive computational runtimes and high storage demands, we introduce a distinct unlearning paradigm that sidesteps inverse Hessian matrix operations while successfully purging designated data classes. Our strategy harnesses the innate forgetting behavior of models during sequential learning, employing a compact subset of data instances as fresh tasks. Following a brief sequence of parameter updates, the framework successfully assists the network in forgetting the target data classes.

The primary contributions of this study are detailed below:

1. We introduce Scrub-and-Learn, an innovative machine unlearning framework that achieves targeted erasure of specific data classes, reinforcing regulatory compliance and privacy protection. In contrast to traditional paradigms, it operates without requiring inverse Hessian matrix operations, access to primary training sets, or the synthesis of supplemental datasets.
2. We present a submaximum one-hot label encoding mechanism tailored for unlearning, which attributes a probability of 1 to the runner-up predicted class and 0 to all alternative positions to signal the elimination of category-specific features.
3. We analyze the cross-category weight-sharing configurations, demonstrating the ways in which neural network parameters contribute to multi-class representations. Furthermore, we investigate the dynamics of catastrophic forgetting during sequential learning, noting how erasing the target category can inadvertently perturb adjacent classes.
4. Comprehensive evaluations across diverse benchmark repositories, including MNIST, FashionMNIST, SVHN, CIFAR-10, and CIFAR-100, reveal that Scrub-and-Learn successfully purges target classes while sustaining the integrity of retained classes. The framework scales robustly across varying dataset volumes and distinct network configurations.

Related Work

Current machine unlearning approaches routinely encounter obstacles linked to heavy computational demands and notable data storage overhead. Our introduced rapid unlearning strategy, directed by a compact selection of samples, draws inspiration from multiple interconnected fields of study.

Machine Unlearning: Ref. [20] leveraged the inverse Hessian matrix calculated across the complete dataset to embed perturbations into the network weights, aiming to strike an equilibrium between model utility and erasure efficacy. Nonetheless, calculating the inverse Hessian matrix is computationally demanding and scales poorly to elaborate

datasets and deep model topologies. To mitigate this issue, ref. [21] recommended saving the inverse Hessian matrix of the kept data as a statistical summary, thereby easing the computational load during the refinement phase. Other strategies approximate the Hessian for individual forgotten points using the mean Hessian evaluated across the entire dataset [22] or substitute it with the Fisher Information Matrix (FIM) [23]. Despite these modifications, the computational burden remains high due to the complex nature of the models and data scales. Ultimately, the most sustainable way to lower computational runtime is to entirely bypass the calculation of the inverse Hessian.

Unlearning approaches that steer clear of Hessian matrix computations typically apply separate loss operations to retained and forgotten partitions, seeking to eliminate features from the forgotten records while preserving the knowledge base of the retained ones. For instance, ref. [27] avoided running parameter updates on the retained data, opting instead to modify only those parameters that exhibited prominent gradients relative to the forgotten data. In a similar fashion, ref. [29] applied gradient descent steps on the retained samples alongside gradient ascent steps on the forgotten samples. Ref. [30] modified feature extraction layers based on the feature space distances of the forgotten records while updating the full set of parameters with the retained data. Nevertheless, these strategies require continuous availability of the primary training sets, expanding storage needs. To alleviate this, certain methods utilize only a minimal fraction of data samples. While this shortens the storage footprint, the network still requires data signals to optimize successfully. Generative models and inversion frameworks are thus deployed to synthesize samples when primary datasets are unavailable. For example, ref. [33, 34] produced adversarial perturbations designed to maximize model prediction error, while [35, 36] applied inversion methodologies to recreate both the retained and the forgotten data instances.

Continual Learning: The ongoing integration of sequential information from nonstationary data distributions frequently triggers catastrophic forgetting or representation interference [38–40]. Catastrophic forgetting [41, 42] describes a sharp degradation in accuracy on previously mastered tasks following the optimization for a novel task. This deterioration of older proficiencies becomes more pronounced as the sequence of incoming tasks lengthens [43]. Several researchers [44, 45] contend that the root cause of catastrophic forgetting stems from a matrix of shared parameters that endows the network with its powerful generalization traits and graceful performance degradation. In pre-trained architectures, distinct category features become increasingly orthogonal as the scale expands, which serves to make the network more resilient against unintended forgetting.

Overcoming catastrophic forgetting represents a foundational hurdle within continual learning research. However, because unlearning is a deliberate objective, we investigate whether the model's inherent susceptibility to forgetting can be purposefully directed to wipe out sensitive details. The core challenge lies in maintaining performance across retained datasets while erasing target sensitive details. Our framework is dedicated to resolving this specific dilemma.

Materials and Methods

In this section, we examine the evolution of network weights throughout the optimization process and analyze the structural sharing of weights among distinct data classes. Subsequently, we present Scrub-and-Learn, a category-aware framework designed to rapidly eliminate class-specific knowledge via strategic network weight modification.

Weight dynamics and category sharing

Neural architectures commonly deployed for classification tasks comprise a feature extractor paired with a downstream classifier. A standard configuration for the classifier involves a fully connected architecture spanning one to three layers. Prior literature [15] thoroughly examines the dynamics of a classifier utilizing a single fully connected layer. Extending this foundation, we investigate classifiers constructed with multiple fully connected layers, centering the following analytical breakdown on a two-layer setup as a representative model.

The classifier receives the feature vector $\hat{x} \in \mathbb{R}^{(d)}$ generated by the feature extractor to compute classification decisions. The activation from the initial layer of the classifier is expressed as $h = \text{ReLU}(\Omega_1 \hat{x} + b_1)$, where $\Omega_1 \in \mathbb{R}^{(m,d)}$ and $b_1 \in \mathbb{R}^{(m)}$

denote the weights and bias vectors of the first connection layer, respectively. The final network output is defined by $\hat{y} = \text{softmax}(\Omega_2 h + b_2)$, where $\Omega_2 \in \mathbb{R}^{(C,m)}$ and $b_2 \in \mathbb{R}^{(C)}$ represent the weights and bias parameters of the output layer, respectively. Throughout training, the classifier minimizes a cross-entropy loss function via gradient descent to maximize classification performance.

Based on previous analytical work [15], during both the starting and advanced phases of optimization, the weight parameters of a single-layer fully connected classifier adjust under the geometric guidance of the Equiangular Tight Frame (ETF), ultimately converging toward the vertices of the ETF, as illustrated in Figure 1a. A classifier converging to an ETF vertex guarantees that the cosine similarity between the row feature vectors of distinct classes approaches $-1/(C-1)$, thereby minimizing their mutual alignment. Consequently, the latent representations of samples belonging to each distinct category within the final layer become as orthogonal as possible, maximizing class separability in the feature space.

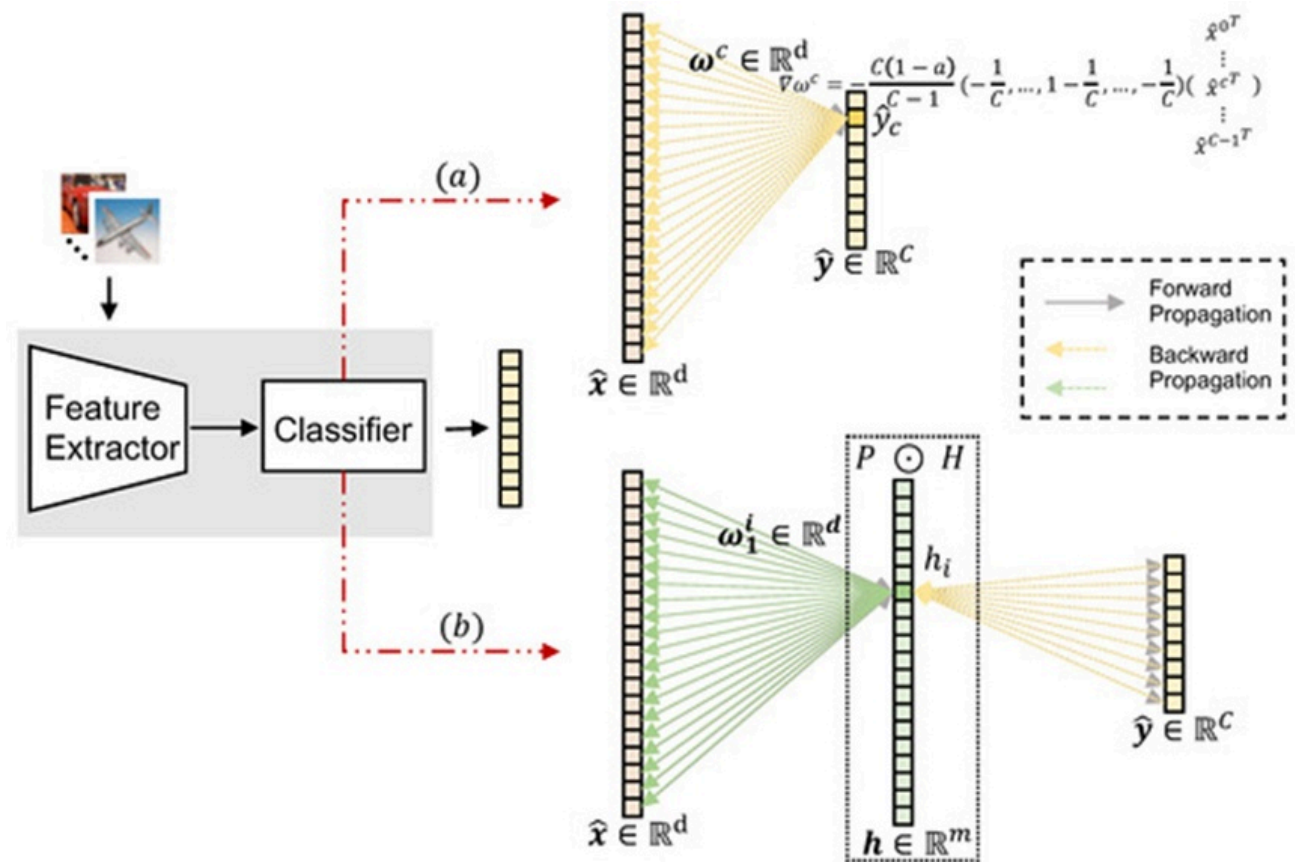


Figure 1. Weights gradients of classifiers with different number of layers. (a) Classifier with one fully connected layer. (b) Classifier with two fully connected layers.

Let the feature vectors extracted from samples across C unique classes be denoted as $X = \{\hat{x}^0, \hat{x}^1, \dots, \hat{x}^{C-1}\} \in \mathbb{R}^{(d,C)}$, where $\hat{x}^c$ represents the extracted feature from a sample of class c , and d indicates the dimensionality of the feature space. After propagating through the classifier, the gradient governing the first-layer weights Ω_1 is formulated in Eq. 1:

$$\nabla \Omega_1 = P \odot H \cdot (A - Y)^T \cdot (A - Y) \cdot \hat{X}^T \quad (1)$$

where P represents an element-wise mask regulating parameter updates, $H=\{h^0, h^1, \dots, h^{C-1}\} \in \mathbb{R}^{(m,C)}$ signifies the first-layer activations corresponding to the classifier input and contains neuron activation states, $A=\{\hat{y}^0, \hat{y}^1, \dots, \hat{y}^{C-1}\} \in \mathbb{R}^{(C,C)}$ denotes the classifier output distributions (softmax probabilities), and $Y=\{y^0, y^1, \dots, y^{C-1}\} \in \mathbb{R}^{(C,C)}$ embodies the one-hot target labels corresponding to $\hat{X}$. This formulation shows that the weight updates in the initial layer are modulated by the output discrepancies $A - Y$ and the feature vectors $\hat{X}$, illustrating the mechanism by which class correlations dictate weight modifications.

This mathematical breakdown establishes that within a two-layer fully connected classifier, the behavior of weight gradients mirrors the single-layer scenario by adjusting class-specific weights according to the underlying ETF geometry. Specifically, the update path for the weight vector ω_1^c of class c aligns with the corresponding row feature vector of class c , while the weights ω_i assigned to alternative classes $i \neq c$ are adjusted in the inverse direction, aligning with prior findings [15]. However, in comparison to the single-layer baseline, the gradient formulation in Eq.1 incorporates an additional modulation term $P \odot H$. This mathematical component indicates that the parameter update trajectory is not solely governed by the ETF configuration but also undergoes an intricate sparse recombination, as shown in Figure 1b. Specifically, $P \odot H$ dictates the manner in which neuron activations and targeted importance values guide weight updates, introducing an extra layer of complexity to the optimization dynamics of multi-layer classifiers.

This finding suggests that even though ETF alignment remains a primary driver for class separation, the integration of $P \odot H$ brings forth an additional selection system that can impact the speed and efficacy of erasing or maintaining specific categories during optimization. Deciphering the influence of this term is essential for engineering streamlined unlearning frameworks that exploit the natural properties of weight evolution within deep classifiers.

We isolate our focus on the weight gradient of the i -th neuron located in the initial layer:

$$\nabla \omega_1^i = \left[-\frac{C(1-a)}{C-1} \right]^2 \cdot p_i^T \odot h_i^T \cdot \left(\mathbf{I} - \frac{1}{C} \mathbf{1}\mathbf{1}^T \right) \cdot \hat{X}^T \quad (2)$$

where $h_i \in \mathbb{R}^C, i=0,1,\dots,m-1$ represents the row vector of H , composed of the i -th component across $\{h^0, h^1, \dots, h^{C-1}\}$. The weight gradient derived in Eq. 2 demonstrates that the update step for each individual neuron depends on a sparse mixture of class features, which are modulated by the joint activation profile $p_i^T \odot h_i^T$. Given that neurons in the initial layer react to shared features spanning separate categories, they naturally fuse information from multiple classes during the backpropagation sequence. Empirical observations from Figure 2 validate this theoretical insight. The confusion matrix generated from MNIST data on a classification model utilizing two fully connected layers reveals that separate blocks of network weights demonstrate distinct class recognition behaviors while highlighting the parameter dependencies shared across multiple categories. Specifically, the weight block in Figure 2b is mainly responsible for identifying class 0, yet it frequently misclassifies other categories (such as 2, 3, 5, 8, and 9) as class 0. Conversely, the weight block in Figure 2c exhibits a high accuracy for identifying class 1, but regularly mistakes classes 2 and 7 for class 1. Both groups of weights exerts a major influence on the classification performance of class 2, indicating that specific weights serve a cross-functional role across multiple classes.

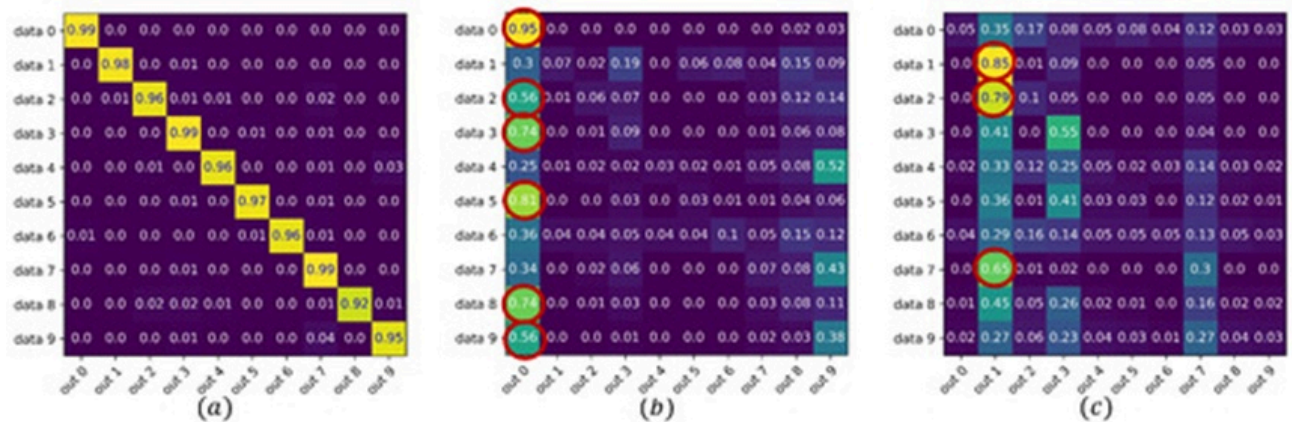


Figure 2. Confusion matrix of MNIST on a classifier. (a) Full model. (b) Only some weights in the first connected layer. (c) Another part of the weights in the first connected layer.

Category-aware weight modification

Deep neural architectures generally develop class discrimination capabilities by minimizing the cross-entropy loss between estimated log probabilities and canonical one-hot target vectors. Under this configuration, the one-hot target acts as a rigid supervision metric, mapping a value of "1" to the correct category index and "0" to all remaining positions. This target encoding dictates the memory retention of the model by regulating the direction of the gradient stream during backward propagation. We posit that substituting the target label of an individual category can systematically steer the architecture to forget that specific class without degrading its capacity to recognize alternative categories.

We introduce Scrub-and-Learn, a category-aware weight modification framework built to erase class-level knowledge from an already optimized model. This strategy reframes the unlearning process through the lens of continual learning, treating the deletion target as a fresh objective function for the pre-trained classification network. It exploits the natural forgetting tendencies of networks undergoing sequential training and employs label-level supervision to drop class-specific information.

To implement this, we engineer a low-overhead yet efficient forgetting routine that directly alters the structure of the one-hot label representations. By pinpointing data instances originating from the class designated for erasure and swapping their target labels with alternative categories derived from the model's current logit outputs, we execute class-level forgetting while preserving the overall training pipeline.

Our routine alters the label assignment step prior to loss estimation, guaranteeing that the model ceases to receive positive gradient reinforcement for the forgotten class. This mechanism, termed submaximum one-hot encoding, maps a value of 1 to the specific index representing the second-highest probability output by the model for the samples slated for deletion, while forcing all other coordinate positions to 0. The complete operational workflow is detailed in [Algorithm 1](#). This architecture maintains the standard training pipeline while provoking targeted forgetting via altered label-level supervision. For instance, in a 10-class classification network tasked with erasing class 0, if the output probability vector is [0.8, 0.01, 0.1, 0.01, 0.01, 0.02, 0.01, 0.01, 0.01, 0.02], the modified target vector becomes [0, 0, 1, 0, 0, 0, 0, 0, 0, 0]. The gradient calculated from this alternative one-hot vector subsequently drives the adjustments within the network weights, as shown in [Figure 3](#).

Require: sample $X \in \mathbb{R}^{N \times \dots}$, ground-truth labels $\mathbf{l} \in \mathbb{N}^N$, forgotten class index c_f , Model $\mathcal{M}$

Ensure: Submaximum one-hot encoding $\mathbf{Y} \in \mathbb{R}^{N \times C}$

- 1: Probability distribution of samples $A \leftarrow \mathcal{M}(X)$
- 2: Identify forgotten-class sample indices: $\mathcal{I} \leftarrow \{i \mid l_i = c_f\}$
- 3: **for** $i \in \mathcal{I}$ **do**
- 4: Remove column c_f : $\tilde{A}_i \leftarrow \text{concat}(A_i[:c_f], \log A_i[c_f + 1 :])$
- 5: Compute predicted alternative class: $l'_i \leftarrow \arg \max_j \tilde{A}_{i,j}$
- 6: Update labels: $l_i \leftarrow l'_i$
- 7: **end for**
- 8: $\mathbf{Y} \leftarrow \text{one_hot}(\mathbf{l}, \text{num_classes} = C)$

Algorithm 1. Submaximum one-hot encoding for forgotten class samples label

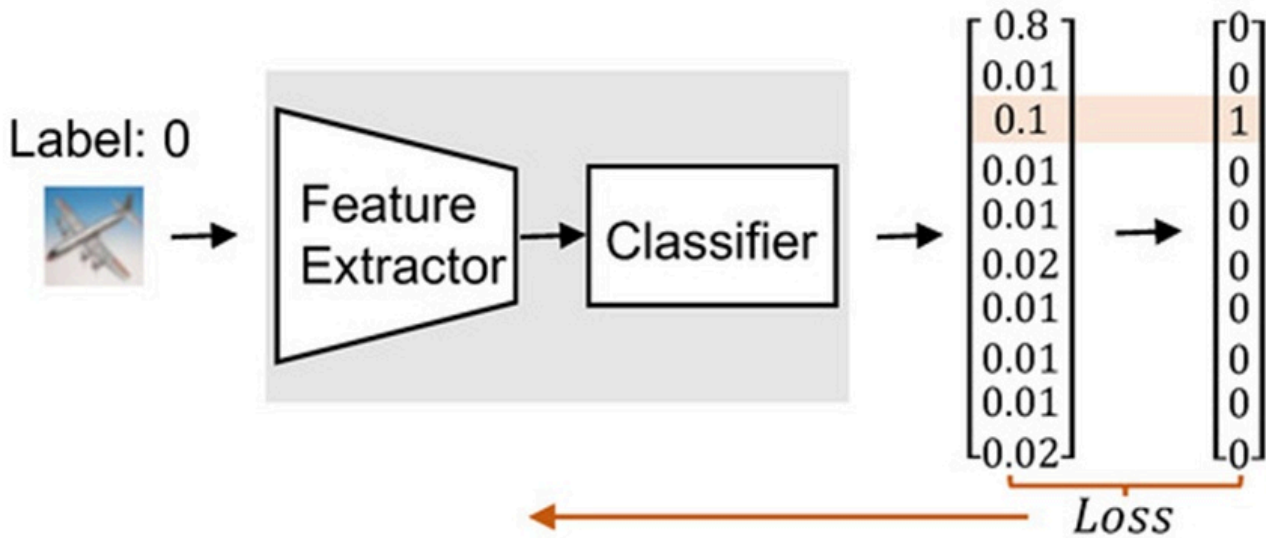


Figure 3. Category-aware weight modification and submaximum one-hot encoding to the labels of forgotten samples.

During the forward pass, the input corresponding to the target sample triggers the activation of weights most closely aligned with it, a process captured by the term $p^{c_f} \odot h^{c_f}$, where c_f denotes the class targeted for removal. The updated label vector acts as the primary supervisory signal—essential for establishing memory within the architecture—and exerts a direct influence on the gradient dynamics that typically reinforce class knowledge during training. In our approach, the original one-hot label of the category to be excised is swapped for a secondary label derived from the model's own predictions. By eliminating the primary supervisory signal, the network is compelled to realign its focus toward other classes. Consequently, the gradient of the model weights is diverted from $\nabla_{\omega} \log \hat{y}_l$ to $\nabla_{\omega} \log \hat{y}_{l'}$, where l' represents the selected alternative class. The model subsequently adjusts its parameters to bolster the representation of this new class rather than the target class, resulting in a progressive attenuation of the original class-specific features. This mechanism exploits the inherent plasticity of neural networks—where sequential learning can effectively overwrite previously stored information—and reformulates the act of forgetting as a directed relearning procedure. Our label-based redirection provides a concise and robust method for category-specific erasure while ensuring the structural integrity of the learning pipeline remains intact.

We also investigated two supplementary encoding strategies for forgetting. The first is random one-hot encoding, where a randomly chosen index in the encoding vector (corresponding to a category not targeted for removal) is set to 1, while all

other indices remain 0. For example, when unlearning class 0, the encoding for a class-0 sample might be expressed as [0, s, s, s, s, s, s, s, s, s], where a randomly selected s is assigned a value of 1. However, this introduces stochasticity into the encoding process, causing the weights to receive inconsistent supervisory signals at each iteration. This also results in a substantial shift in the model's loss function, expressed in Eq. 3:

$$L = - \sum_c y_c \log(\hat{y}_c) = - \log(\hat{y}_s) \geq - \log(\hat{y}_r) \quad (3)$$

As a result, the model's accuracy on the retained dataset degrades. The second method is all-zero encoding, where the supervisory signal for the target category is a vector containing only zeros. For instance, in this configuration, samples from class 0 are encoded as [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]. However, this results in a constant loss of 0, as defined in Eq. 4:

$$L = - \sum_c \hat{y}_c \log(\hat{y}_c) = 0 \quad (4)$$

This eliminates the learning signal entirely, rendering the model incapable of successfully forgetting the target data.

As illustrated in Figure 4, the accuracy of VGG16 on CIFAR-10 following the unlearning process is presented separately for retained data (shown in the top portion of each subimage) and forgotten data (shown in the bottom portion). The horizontal axis k represents the cumulative number of iterations, the vertical axis corresponds to batch size, and the color gradient represents the mean accuracy across 30 trials. Each subimage corresponds to a distinct encoding strategy: (a) submaximum one-hot encoding effectively purges class 0 while maintaining performance on retained data; (b) random one-hot encoding successfully forgets the target but simultaneously impairs accuracy on retained classes; and (c) all-zero encoding fails to clean the class 0 data, with accuracy remaining at approximately 60% after 10 iterations, demonstrating its ineffectiveness.

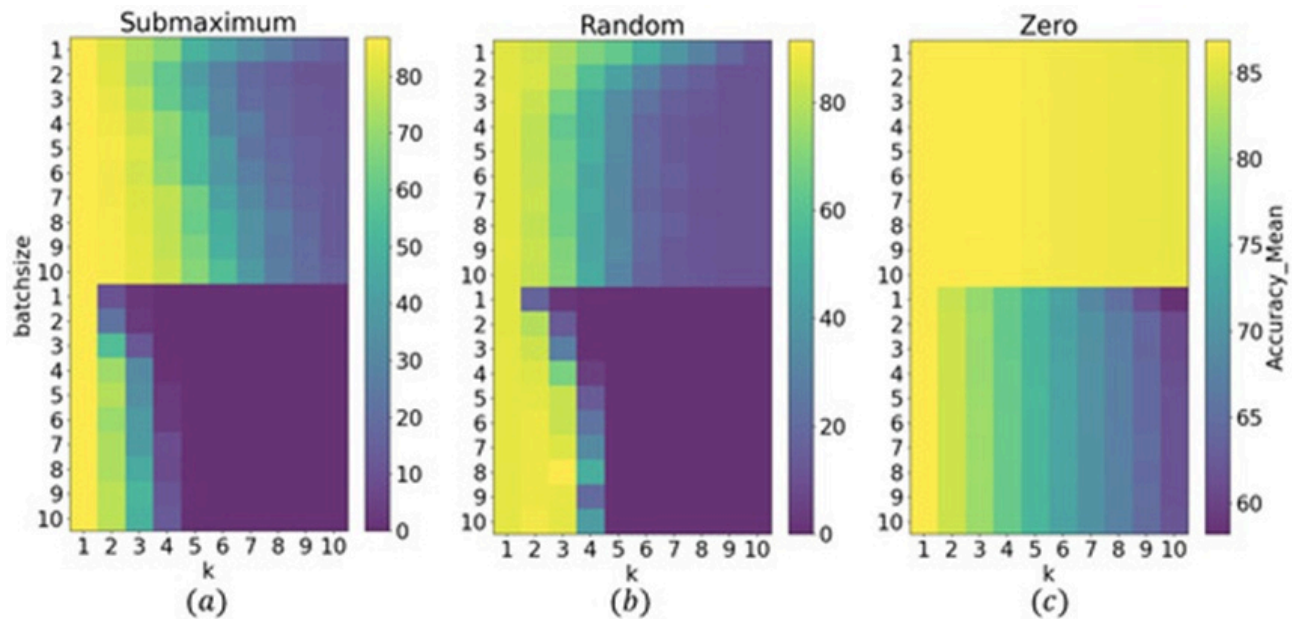


Figure 4. Network performance on the forgetting task using different encoding strategies: (a) submaximum one-hot encoding, (b) random one-hot encoding, and (c) all-zero encoding.

Compared to entropy maximization or techniques based on adversarial perturbations, our method ensures stable unlearning with minimal disruption to existing knowledge by selectively replacing labels with the second-highest predicted

category. This approach prevents the introduction of excessive volatility or instability into the neural network. Furthermore, unlike adversarial strategies that depend on intricately constructed noise, our approach is straightforward, efficient, and less prone to compromising general model performance.

Challenges in the unlearning process

Because network weights are shared across diverse categories, samples from the forgotten class primarily modify the parameters most relevant to them. However, as the network learns subsequent tasks, these shared weights undergo continuous adjustment, which can lead to the catastrophic forgetting of retained data.

We experimentally replicated the one-epoch forgetting sequence across various datasets and architectures. A new task requires the pre-trained classifier to purge class-0 samples. To assess the unlearning behavior, we partitioned the test dataset into two segments: a "forgetting set" consisting exclusively of class-0 samples and a "retention set" consisting of samples from all other classes. The total number of iterations k is defined as $k = \text{Num_samples} / \text{batchsize}$, where Num_samples denotes the total quantity of samples utilized during the unlearning task. In these experiments, both k and batchsize varied from 1 to 10, with the exception of the CIFAR100 dataset, where k took values from the set {3, 6, 9, 12, 15, 18, 21, 24, 27, 30}. In the i -th trial, the sample index range was $[i \cdot 10, (i+1) \cdot 10 + \text{Num_samples}]$. For every batch size and iteration count, classification accuracy was averaged over 30 independent runs to enhance robustness and ensure comprehensive sample coverage. **Figure 5** illustrates the complete experimental results, where each subgraph corresponds to a specific model–dataset pair. In each subgraph, the top panel depicts accuracy on the retention set, while the bottom panel depicts accuracy on the forgotten set, with the axes and color scale representing iteration count, batch size, and accuracy levels, respectively. As previously noted, networks trained sequentially typically undergo catastrophic forgetting within a single cycle. Notably, the model displays a sequential forgetting dynamic: it discards samples from the forgetting set first, followed by a gradual reduction in accuracy for the retained set.

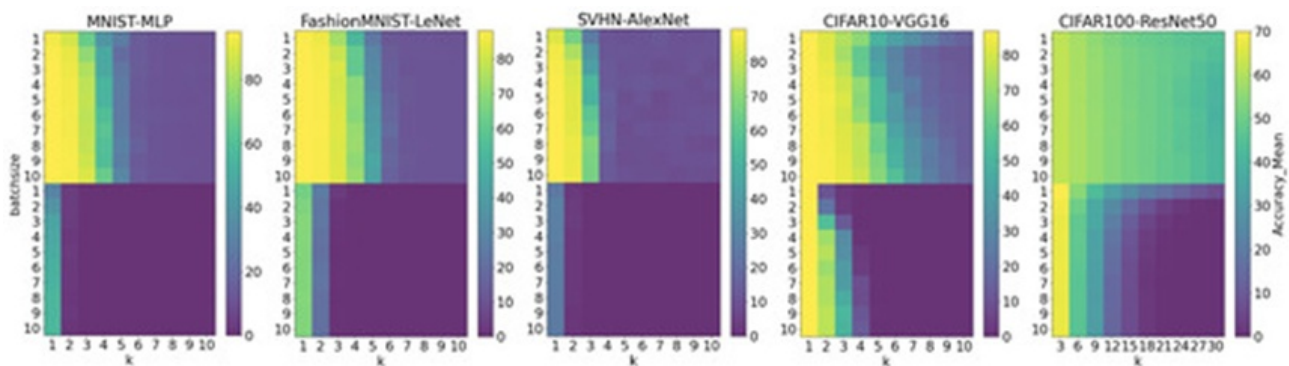


Figure 5. The performance of the model after continued learning on different data–model pairs. The forgotten set data are forgotten first, and then the retained set data are forgotten.

Consequently, when a model unlearns a specific category in a continual learning context, the primary challenge is to effectively balance targeted data removal with the preservation of performance on retained classes. It is critical to terminate the unlearning task before catastrophic forgetting begins to compromise the retention set, thereby maintaining an optimal equilibrium between forgetting and retention.

Sample selection for unlearning

As a model proceeds with ongoing training, any significant divergence in the new objective—such as moving from identifying cat breeds to categorizing bird species—requires a training set for the new iteration that is equal to or larger than the previous one. These inputs are critical for the model to successfully adapt to the new requirement. Within a continual learning paradigm, the deletion of a specific class is treated as the inception of a fresh task that omits that category. For instance, if the goal is to purge class 0 while keeping the classification logic for all other classes intact,

those remaining classifications are viewed as a holdover from the original task. The inherent connection between the original and subsequent tasks suggests that the new training inputs are effectively drawn from the original dataset. Moreover, to avoid "catastrophic forgetting"—the loss of previously acquired knowledge—the number of training cycles must be constrained; consequently, only a sparse set of samples is required to execute the unlearning process effectively.

Empirical testing suggests that the ideal sample size for a new task should not exceed $5C$, where C represents the total number of classes. Our trials indicated that deploying smaller batch sizes accelerated the unlearning procedure. Specifically, batch sizes beneath 10 for a 10-class dataset and beneath 20 for a 100-class dataset yielded optimal results. Conversely, when the iteration limit surpassed 5 for the 10-class case or 20 for the 100-class case, the model's proficiency on the retained data declined sharply. Thus, these thresholds serve as recommended heuristic guidelines.

If the algorithm causes the model's retention accuracy to dip when trained solely on the targeted "forgotten" samples, incorporating a subset of retained class samples can counteract this drop. **Figure 6** demonstrates how, with VGG16 continuing its training on CIFAR-10 using a consistent total sample count, the composition of the sample categories influences performance. The X-axis represents the aggregate number of samples. The figure shows (a) containing only the deleted category, while (b) integrates representative samples from the preserved classes. Red markers highlight that the model can successfully erase class 0 when the total sample size is 13 and the batch size is 4. Including samples from the retained classes clearly stabilizes the model's retention performance.

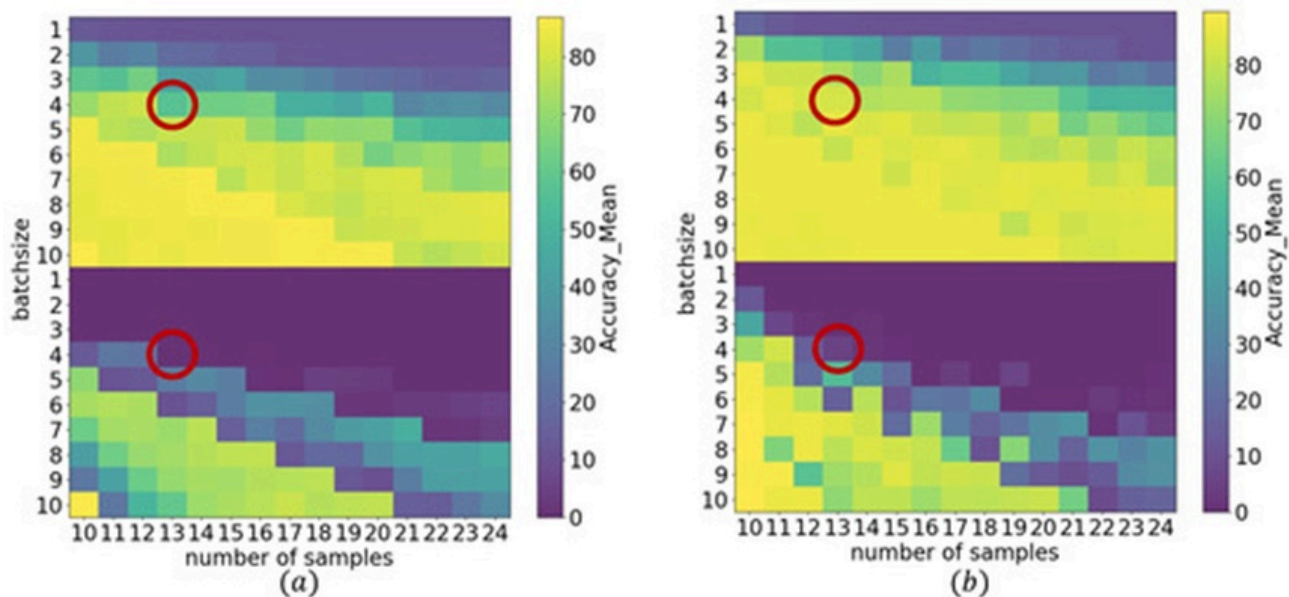


Figure 6. The forgetting performance of VGG16 under different training sample sizes: (a) includes only the forgotten class samples; (b) includes both the forgotten and retained class samples.

Experiment

This section delineates the experimental configuration, the operational workflow, and the finalized outcomes. Our primary ambition is to confirm the efficacy of the proposed mechanism and benchmark it against current high-performance methods across various datasets.

Experimental setting

We tailored the network architectures to match the complexity of the tasks associated with each dataset. For MNIST [46], we utilized MLP and LeNet [46] classifiers. Regarding FashionMNIST [47], the testing involved MLP, LeNet, and AlexNet [48] models. For SVHN [49], we deployed AlexNet, VGG11 [50], and ResNet18 [51] networks. The CIFAR10 [52] assessments utilized VGG16, ResNet34, InceptionV3 [53], and ViT-S [54] classifiers. For the CIFAR100 dataset, we integrated VGG16, ResNet50, InceptionV3, and ViT-S architectures.

To test the robustness of our approach in privacy-sensitive domains, we extended our experiments to the Augmented Skin Conditions Image (ASCI) dataset [55], which consists of clinical dermatological images spanning six distinct conditions.

All models were developed in PyTorch and trained via stochastic gradient descent (SGD). To provide a clear baseline, we trained each network from scratch using standardized hyperparameter settings. We established momentum at 0.9, weight decay at 0.0005, and an initial learning rate of 0.1, utilizing CosineAnnealingLR for dynamic adjustment. Models for MNIST, FashionMNIST, and SVHN were trained over 100 epochs with a batch size of 256. CIFAR10 and CIFAR100 models were trained for 200 epochs with a batch size of 128. Once baselines were established, we applied our sample-guided fast unlearning procedure to excise a specific class.

For single-class trials, class 0 was chosen as the target for removal. On MNIST, FashionMNIST, and SVHN, we randomly sampled eight instances of class 0, utilizing a learning rate of 0.01 and a batch size of 4. For CIFAR-10, we combined one sample from each preserved class with four samples of the forgotten class, using a learning rate of 0.05 and the same batch size. For CIFAR-100, one sample per retained class and ten from the target class were used, with a learning rate of 0.2 and a batch size of 16. For multi-class trials, we targeted classes 0 and 4 in the 10-class dataset and 20 randomly selected classes for the 100-class dataset. For MNIST, FashionMNIST, and SVHN, we drew eight samples per forgotten class at learning rates of 0.01 and batch sizes of 8, 6, and 2, respectively. CIFAR-10 used one sample from each retained class and four per forgotten class (LR 0.05, batch size 4), while CIFAR-100 used one sample per retained class and four per forgotten class (LR 0.5, batch size 16). All optimizer settings mirrored the original training phase.

The computational infrastructure consisted of a Linux server featuring two Intel® Xeon® Silver 4214R CPUs (24 cores/48 threads), 440 GB RAM, and an NVIDIA GeForce RTX 3080 Ti GPU (12 GB VRAM). The environment was configured with Python 3.8, PyTorch 1.10.0 (CUDA 11.3/cuDNN 8.2), and Ubuntu 20.04 LTS.

Evaluation of unlearning networks

Tables 1 and 2 detail the accuracy metrics for both baseline and unlearning models regarding preserved versus removed data. By selecting 30 sample groups via the aforementioned indexing technique, we captured the peak accuracy for each set. The findings in **Table 1** reveal that our algorithm maintained strong performance across diverse architectures for single-class removal. In every case, the "forgetting accuracy" stayed below 5%, with the variance between original and post-unlearning accuracy on retained data remaining under 4%. **Table 2** indicates that erasing two categories in a 10-class dataset produces results parallel to single-class removal. However, removing 20 categories from the 100-class dataset caused a more noticeable dip in retained accuracy. This likely stems from the higher volume of parameter updates required for broader removals, which can inadvertently compromise the model's overall generalization capability.

Table 1. The performance of single-class unlearning evaluated across multiple datasets and various networks.

Dataset	Architecture	$\Delta\text{Acc_R}$ (%) $\uparrow$	Unlearned Model Accuracy on Forgotten Data Acc_F (%) $\downarrow$	Unlearned Model Accuracy on Retained Data Acc_R (%)	Baseline Model Accuracy on Forgotten Data Acc_F (%)	Baseline Model Accuracy on Retained Data Acc_R (%)
MNIST	MLP	-0.45	1.22	98.04	99.39	98.49

	LeNet	−0.52	0.00	98.57	99.59	99.09
FashionMNIST	MLP	−1.35	1.30	89.14	86.20	90.49
	LeNet	−0.65	2.00	90.18	87.50	90.83
	AlexNet	−2.66	2.20	89.84	87.50	92.50
SVHN	AlexNet	−1.24	4.58	92.73	94.38	93.97
	VGG11	−0.36	0.23	95.32	96.62	95.68
	ResNet18	−1.77	3.15	94.40	96.62	96.17
CIFAR10	VGG16	−0.02	0.00	89.50	92.80	89.52
	ResNet34	−1.77	0.70	87.33	91.00	89.10
	InceptionV3	−1.40	4.90	91.91	93.80	93.31
	ViT-S	−2.65	3.10	93.19	97.10	95.84
CIFAR100	VGG16	−0.54	0.00	64.45	86.00	64.99
	ResNet50	−1.45	1.00	64.60	88.00	66.05
	InceptionV3	−2.98	0.00	72.17	88.00	75.15
	ViT-S	−3.52	0.00	79.26	92.00	82.78
ASCI	ResNet50	−3.75	2.53	94.25	92.41	98.00
	ViT-S	−2.75	3.80	94.25	94.94	97.00

Table 2. The performance of multi-class unlearning evaluated across multiple datasets and various networks.

Dataset	Network Architecture	Change in Retained Accuracy ΔAcc_R (%) ↑	Unlearned Model Accuracy on Forgotten Samples Acc_F (%) ↓	Unlearned Model Accuracy on Retained Samples Acc_R (%)	Baseline Model Accuracy on Forgotten Samples Acc_F (%)	Baseline Model Accuracy on Retained Samples Acc_R (%)
MNIST	MLP	−1.08	4.17	97.44	98.83	98.52
	LeNet	−1.84	3.56	97.20	99.54	99.04
FashionMNIST	MLP	0.00	3.60	91.23	85.40	91.23
	LeNet	−0.03	4.20	91.68	84.60	91.71
SVHN	VGG11	−2.05	0.02	94.25	95.62	96.31
	ResNet18	−0.57	1.89	95.50	96.86	96.07
CIFAR10	VGG16	0.45	0.44	89.91	91.40	89.46
	ResNet34	−1.42	3.50	87.74	89.80	89.16

CIFAR100	VGG16	−13.94	0.10	50.66	67.60	64.60
	ResNet50	−8.25	4.65	57.59	68.00	65.84

Significantly, the medical imaging data (Table 1, bottom rows) confirms that our method successfully balances rapid unlearning with the preservation of classification accuracy on retained content, validating its utility for privacy-focused real-world applications.

Comparison with other unlearning methods

Table 3 evaluates the proposed framework against contemporary state-of-the-art techniques, including PBU [7], GKT [33], WF-Net [31], and NG-IR [34], utilizing the MNIST, SVHN, CIFAR-10, and CIFAR-100 datasets. Whereas alternative methodologies triggered a substantial degradation in accuracy on the retained datasets, our strategy minimized the adverse effects on the model's preservation performance. All benchmark models were analyzed under an identical class-level erasure task, utilizing the performance metrics reported in their respective original publications.

Table 3. Comparison of single-class unlearning with state-of-the-art methods on MNIST, SVHN, CIFAR10, and CIFAR100.

Dataset	Approach	Architecture	Change in Retained Accuracy $\Delta\text{Acc_R}$ (%) ↑	Post-Unlearning Accuracy on Forgotten Data Acc_F (%) ↓	Post-Unlearning Accuracy on Retained Data Acc_R (%)	Baseline Accuracy on Forgotten Data Acc_F (%)	Baseline Accuracy on Retained Data Acc_R (%)
MNIST	PBU	ResNet18	−2.76	0.21	96.24	99.19	99.00
		AlICNN	−1.09	0.07	98.40	99.37	99.49
		ResNet34	−3.43	0.01	96.02	99.43	99.45
	GKT (zero-shot)	AlICNN	−0.72	0.00	97.12	99.61	97.84
		LeNet	−2.36	0.00	95.79	99.59	98.15
		ResNet9	−4.00	0.00	94.57	99.10	98.57
	WF-Net	VGG16	−26.40	0.00	73.20	99.60	99.60
		ResNet18	−5.60	9.68	94.00	99.60	99.60
		ViT-T	−5.40	0.00	93.50	98.90	98.90
	Proposed Method	MLP	−0.45	1.22	98.04	99.39	98.49
SVHN	GKT (zero-shot)	LeNet	−0.52	0.00	98.57	99.59	99.09
		AlICNN	−2.09	0.00	92.43	95.16	94.52

		LeNet	-7.42	0.00	78.27	81.42	85.69
		ResNet9	-43.32	0.00	39.44	87.11	82.76
	Proposed Method	AlexNet	-1.24	4.59	92.73	94.38	93.97
		VGG11	-0.36	0.23	95.32	96.62	95.68
		ResNet18	-1.77	3.15	94.40	96.62	96.17
CIFAR10	PBU	ResNet18	-10.38	4.50	66.16	71.82	76.54
		AIICNN	-8.77	1.04	76.15	79.64	84.92
		ResNet34	-8.10	0.59	68.61	68.52	76.78
	GKT (zero-shot)	AIICNN	-12.08	0.00	81.97	87.49	94.05
		LeNet	-18.48	0.00	41.32	62.25	59.80
		ResNet9	-28.00	0.00	56.83	88.50	84.83
	NG-IR	ResNet18	-6.26	0.00	71.60	81.01	77.86
		AIICNN	-8.74	0.00	73.90	91.02	82.64
	WF-Net	VGG16	-12.80	18.30	80.20	93.00	93.00
		ResNet18	-14.20	9.25	79.70	94.00	93.90
		ViT-T	-4.50	0.00	73.50	78.00	78.00
	Proposed Method	VGG16	-0.02	0.00	89.50	92.80	89.52
		ResNet34	-1.77	0.70	87.33	91.00	89.10
		InceptionV3	-1.40	4.90	91.91	93.80	93.31
		ViT-S	-2.65	3.10	93.19	97.10	95.84
CIFAR100	PBU	ResNet18	-6.51	1.50	69.55	80.11	76.06
		ResNet50	-6.66	0.33	69.29	78.44	75.95
		ResNet34	-9.87	0.17	65.34	84.00	75.21
	NG-IR	ResNet18	-3.32	0.00	75.36	83.00	78.68
		MobileNetV2	-1.67	0.00	75.76	90.00	77.43
	WF-Net	VGG16	-12.80	18.30	80.20	93.00	93.00
		ResNet18	-14.20	9.25	79.70	94.00	93.90
		ViT-T	-4.50	0.00	73.50	78.00	78.00
	Proposed Method	VGG16	-0.55	0.00	64.44	86.00	64.99
		ResNet50	-1.46	1.00	64.60	88.00	66.05

		InceptionV3	−2.98	0.00	72.17	88.00	75.15
		ViT-S	−3.52	0.00	79.26	92.00	82.78

Analyze the impact of learning rate on the unlearning methods

This analysis investigated how varying the learning rate affects the unlearning algorithm, using a VGG16 classifier trained on the CIFAR10 dataset. During the eradication of class 0, all environmental variables were held constant while the learning rate was incrementally adjusted from 0.01 to 0.1, with a step size of 0.01 (0.01, 0.02, 0.03, \dots, 0.1). **Figure 7** demonstrates the unlearning performance observed across these different learning rates. The choice of learning rate primarily dictated the number of optimization steps required to achieve erasure; specifically, elevated learning rates corresponded to a reduced number of iterations. In practical deployment, selecting an appropriate learning rate necessitates navigating a delicate trade-off between complete data erasure and knowledge retention.

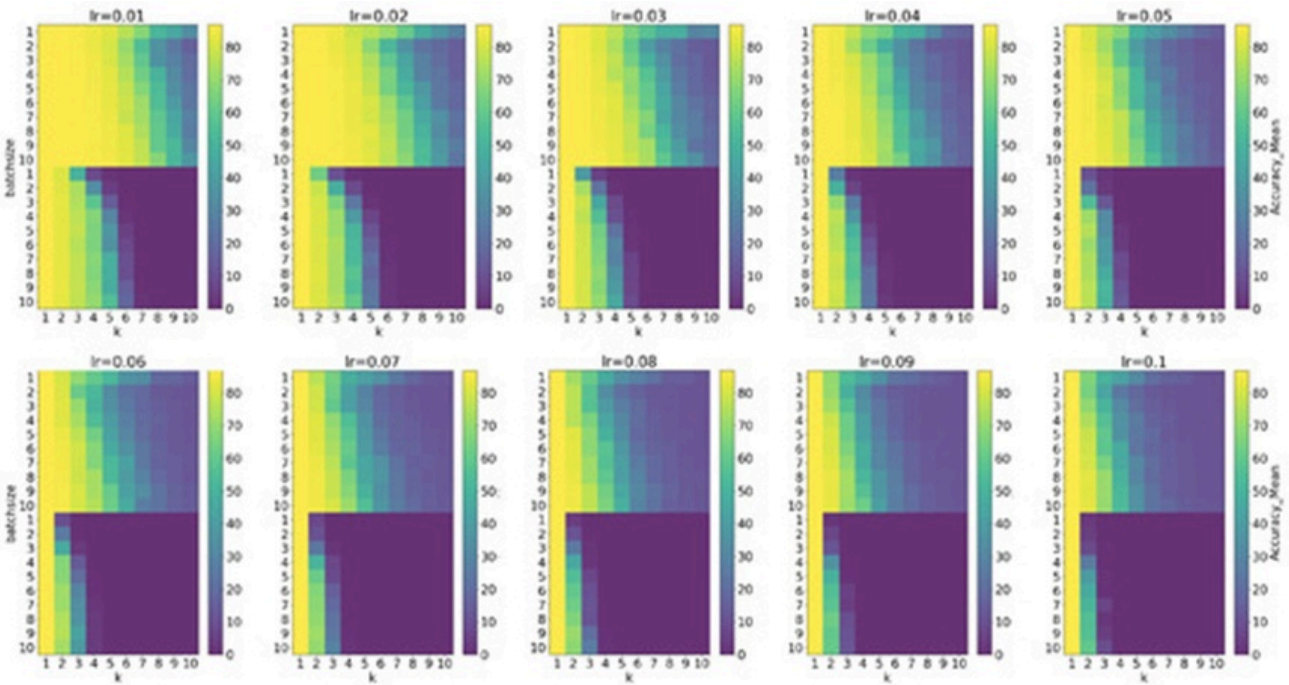


Figure 7. The performance of the model after forgetting learning with different learning rates on CIFAR10.

T-SNE visualization of the classification results of the unlearning model

This experiment explored the structural changes induced by our unlearning method on LeNet architectures trained on the MNIST dataset. By applying the intended erasure pipeline, we generated two distinct unlearning variants: a single-class forgetting model (targeting class 0 exclusively) and a multi-class forgetting model (targeting both class 0 and class 4). **Figure 8** maps the classification boundaries of the baseline model (a), the single-class unlearning variant (b), and the multi-class unlearning variant (c) via t-SNE dimensionality reduction. The visualizations reveal that the proposed erasure method forces the model to reassign data from the forgotten category to alternative classes. This is achieved by systematically suppressing the second-highest confidence encoding within the sample labels, thereby shifting the model's comprehensive decision boundaries as highlighted by the black dashed circle.

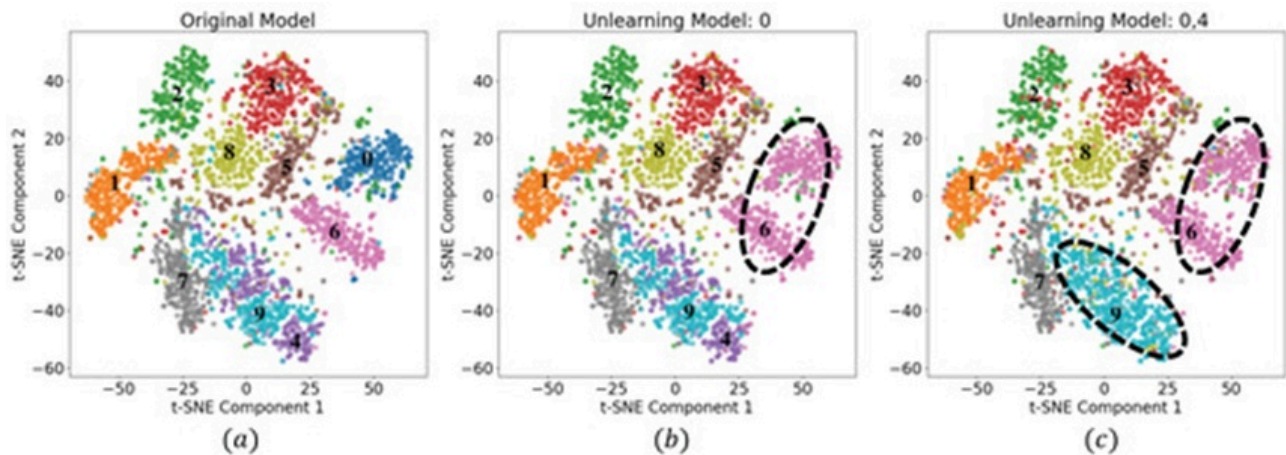


Figure 8. t-SNE visualization of the classification results of the LeNet model on the MNIST dataset: (a) original model, (b) unlearning class 0, (c) unlearning class 0 and 4.

Results and Discussion

Method analysis

This section contextualizes our findings within the broader landscape of existing unlearning techniques, examining both the structural benefits and the constraints of the proposed architecture. In comparison to influence-based erasure techniques [20–23] and sample-heavy methodologies [26–32], our protocol provides a straightforward and potent alternative for class-targeted unlearning.

Influence-based paradigms depend on second-order statistical profiles or influence functions to determine parameter sensitivity. While rigorous, these strategies incur steep computational costs due to the necessity of calculating or approximating the Fisher Information Matrix or Jacobian–Hessian products, which generally restricts their deployment to shallow networks. Conversely, our approach circumvents this computational bottleneck by operating directly within the label space—substituting the true label with the model's secondary prediction choice—to trigger erasure. This architecture streamlines the unlearning phase by removing the necessity for second-order derivative tracking and complex retraining loops.

When contrasted with retraining-centric frameworks like SISA [56], which demand structural network partitioning and the sequential retraining of isolated submodels, our method leverages a lightweight fine-tuning pipeline governed entirely by altered label inputs. This design mitigates hardware strain and increases compatibility with deep network models and heterogeneous data structures.

Data-driven unlearning frameworks typically apply gradient ascent or loss maximization strategies to force data omission during fine-tuning or retraining phases. However, these paradigms require access to the comprehensive original dataset or necessitate the creation of massive synthetic data pools. Our framework markedly lowers these data volume requirements. Furthermore, unlike strategies that demand a complete training reset or prolonged fine-tuning intervals, our methodology delivers an efficient, plug-and-play unlearning mechanism for pre-trained networks.

Empirical verification across multiple distinct benchmarks confirms that our method reliably drives classification accuracy for targeted classes down to $\leq 5\%$ while restricting the accuracy drop on retained classes to $\leq 4\%$.

Limitations and future work

Despite encouraging validation metrics, the proposed strategy has certain limitations. First, the framework is specialized for class-level erasure and lacks support for instance-level or feature-level unlearning, which are crucial for granular data-removal scenarios. Second, the algorithm's performance deteriorates when tasked with the concurrent removal of multiple distinct classes. Expanding the volume of target classes introduces a wider array of affected parameters, which can noticeably degrade the model's systemic performance. Third, this research centers on the empirical evaluation of unlearning performance and lacks formalized, provable privacy guarantees. These dimensions require deeper investigation.

In subsequent research, we intend to mitigate these challenges by adapting our framework to handle multi-class and sequential unlearning tasks, where multiple categories are expunged from the network dynamically or consecutively. We also plan to refine the methodology by integrating parameter-importance metrics to boost both accuracy and interpretability. Additionally, we aim to investigate adaptive label-substitution methods that better preserve inter-class correlations, alongside establishing theoretical boundaries for data erasure. Finally, we intend to test the viability of our framework in broader landscapes, such as large language models and privacy-critical commercial software.

Conclusion

In this study, we introduced Scrub-and-Learn, an innovative unlearning framework that successfully erases knowledge linked to specific data classes without requiring complex Hessian inverse computations. By exploiting submaximum one-hot encodings and utilizing a highly constrained data footprint, Scrub-and-Learn forces the network to forget target categories while maintaining structural integrity over preserved data classes. Comprehensive experimentation across numerous datasets and network topologies establishes the efficiency and potency of Scrub-and-Learn within single-class unlearning paradigms. These findings underscore the functional utility of our framework as a secure, precise, and scalable solution for deep model unlearning.

Abbreviations

The following abbreviations are used in this manuscript:

FIM	Fisher Information Matrix
ETF	Equiangular Tight Frame
RAM	Random-access memory
GPU	Graphics Processing Unit
VRAM	Video RAM
CUDA	Compute Unified Devices Architecture
PBU	Partially Blinded Unlearning
GKT	Gated Knowledge Transfer
WF-Net	Weight Filtering-Net
NG-IR	Noise Generation-Impair Repair
ASCI	Augmented Skin Conditions Image

Acknowledgements

None

Conflict of interest

None

Financial support

This work is supported in part by Beijing Natural Science Foundation-Joint Funds of Haidian Original Innovation Project (Grant No. L232056), Science and Technology Innovation 2030 Major Projects (Grant No. 2022ZD0211603).

Ethics statement

None

Received: 15 Sep 2022

Revised: 14 Oct 2022

Accepted: 26 Nov 2022

Published online: 15 March 2023

Rights and permissions

Open Access The author(s) retain copyright. This article is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. It may be shared and adapted for non-commercial purposes with appropriate attribution, an indication of changes, and distribution of adaptations under the same license. Third-party material may be subject to separate terms identified in its credit line. View the license at <https://creativecommons.org/licenses/by-nc-sa/4.0/>.

References

- Xu J, Wu Z, Wang C, Jia X. Machine unlearning: solutions and challenges. *IEEE Trans Emerg Top Comput Intell*. 2024;8(3):2150-68.
<https://doi.org/10.1109/TETCI.2024.3379240>.
- Zhang H, Nakamura T, Isohara T, Sakurai K. A review on machine unlearning. *SN Comput Sci*. 2023;4(4):337.
<https://doi.org/10.1007/s42979-023-01767-4>.
- Shaik T, Tao X, Xie H, Li L, Zhu X, Li Q. Exploring the landscape of machine unlearning: a comprehensive survey and taxonomy. *IEEE Trans Neural Netw Learn Syst*. 2025;36(7):11676-96.
<https://doi.org/10.1109/TNNLS.2024.3486109>.
- Cevallos ID, Benalcázar ME, Valdivieso Caraguay ÁL, Zea JA, Barona-López LI. A systematic literature review of machine unlearning techniques in neural networks. *Computers*. 2025;14(4):150.
<https://doi.org/10.3390/computers14040150>.
- Li N, Zhou C, Gao Y, Chen H, Zhang Z, Kuang B, et al. Machine unlearning: taxonomy, metrics, applications, challenges, and prospects. *IEEE Trans Neural Netw Learn Syst*. 2025;36(8):13709-29.
<https://doi.org/10.1109/TNNLS.2025.3530988>.

De Min T, Mancini M, Lathuilière S, Roy S, Ricci E. Unlearning personal data from a single image. *Trans Mach Learn Res*. 2025. Available from: OpenReview.

Panda S, Sourav S, A.P. P. Partially blinded unlearning: class unlearning for deep networks from Bayesian perspective. *Proc AAAI Conf Artif Intell*. 2025;39(6):6372-80.
<https://doi.org/10.1609/aaai.v39i6.32682>.

Chen K, Huang Y, Wang Y, Zhang X, Mi B, Wang Y. Privacy preserving machine unlearning for smart cities. *Ann Telecommun*. 2024;79:61-72.

Chen K, Wang Z, Mi B. Private data protection with machine unlearning in contrastive learning networks. *Mathematics*. 2024;12(24):4001.
<https://doi.org/10.3390/math12244001>.

Wei S, Zhang M, Zha H, Wu B. Shared adversarial unlearning: backdoor mitigation by unlearning shared adversarial examples. *Adv Neural Inf Process Syst*. 2023;36:25876-909.

Guo Y, Zhao Y, Hou S, Wang C, Jia X. Verifying in the dark: verifiable machine unlearning by using invisible backdoor triggers. *IEEE Trans Inf Forensics Secur*. 2024;19:708-21.
<https://doi.org/10.1109/TIFS.2023.3328269>.

European Parliament, Council of the European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council (General Data Protection Regulation) [Internet]. 2016 [cited 2024 Oct 14]. Available from: EUR-Lex.

Zhang D, Pan S, Hoang T, Xing Z, Staples M, Xu X, et al. To be forgotten or to be fair: unveiling fairness implications of machine unlearning methods. *AI Ethics*. 2024;4:83-93.

Chen R, Yang J, Xiong H, Bai J, Hu T, Hao J, et al. Fast model debias with machine unlearning. *Adv Neural Inf Process Syst*. 2023;36:14516-39.
<https://doi.org/10.52202/075280-0639>.

Wang J, Bie H, Jing Z, Zhi Y, Fan Y. Weight masking in image classification networks: class-specific machine unlearning. *Knowl Inf Syst*. 2025;67:1-21.
<https://doi.org/10.1007/s10115-024-02312-2>.

Dolatabadi HM, Erfani SM, Leckie C. Adversarial coreset selection for efficient robust training. *Int J Comput Vis*. 2023;131:3307-31.

Maalouf A, Eini G, Mussay B, Feldman D, Osadchy M. A unified approach to coreset learning. *IEEE Trans Neural Netw Learn Syst*. 2024;35(5):6893-905.
<https://doi.org/10.1109/TNNLS.2022.3213169>.

Huang Y, Yuan X, Wang H, Du Y. Coreset selection can accelerate quantum machine learning models with provable generalization. *Phys Rev Appl*. 2024;22(1):014074.

Zhao P, Zhang K, Zhang H, Chen H. Alternating minimization differential privacy protection algorithm for the novel dual-mode learning tasks model. *Expert Syst Appl*. 2025;259:125279.
<https://doi.org/10.1016/j.eswa.2024.125279>.

Guo C, Goldstein T, Hannun A, Van Der Maaten L. Certified data removal from machine learning models. In: *Proceedings of the 37th International Conference on Machine Learning*. 2020;119:3832-42.

Sekharia A, Acharya J, Kamath G, Suresh AT. Remember what you want to forget: algorithms for machine unlearning. *Adv Neural Inf Process Syst*. 2021;34:18075-86.

Suriyakumar V, Wilson AC. Algorithms that approximate data removal: new results and limitations. *Adv Neural Inf Process Syst*. 2022;35:18892-903.
<https://doi.org/10.52202/068431-1372>.

Peste A, Alistarh D, Lampert CH. SSSE: efficiently erasing samples from trained machine learning models. In: *NeurIPS 2021 Workshop on Privacy in Machine Learning*; 2021.

Zhang Y, Lu Z, Zhang F, Wang H, Li S. Machine unlearning by reversing the continual learning. *Appl Sci*. 2023;13(16):9341.
<https://doi.org/10.3390/app13169341>.

Mahadevan A, Mathioudakis M. Certifiable unlearning pipelines for logistic regression: an experimental study. *Mach Learn Knowl Extr*. 2022;4(3):591-620.
<https://doi.org/10.3390/make4030029>.

Nguyen QP, Low BKH, Jaillet P. Variational Bayesian unlearning. *Adv Neural Inf Process Syst*. 2020;33:16025-36.

Fan C, Liu J, Zhang Y, Wong E, Wei D, Liu S. SalUn: empowering machine unlearning via gradient-based weight saliency in both image classification and generation. *arXiv [Preprint]*. 2023;arXiv:2310.12508.

Kurmanji M, Triantafillou P, Hayes J, Triantafillou E. Towards unbounded machine unlearning. *Adv Neural Inf Process Syst*. 2023;36:1957-87.

Trippa D, Campagnano C, Bucarelli MS, Tolomei G, Silvestri F. ∇ t: gradient-based and task-agnostic machine unlearning. *arXiv [Preprint]*. 2024;arXiv:2403.14339.

Cotogni M, Bonato J, Sabetta L, Pelosin F, Nicolosi A. DUCK: distance-based unlearning via centroid kinematics. *arXiv [Preprint]*. 2023;arXiv:2312.02052.

Poppi S, Sarto S, Cornia M, Baraldi L, Cucchiara R. Multi-class unlearning for image classification via weight filtering. *IEEE Intell Syst*. 2024;39(6):40-7.
<https://doi.org/10.1109/MIS.2024.3412742>.

Wang W, Zhang C, Tian Z, Yu S. Machine unlearning via representation forgetting with parameter self-sharing. *IEEE Trans Inf Forensics Secur*. 2024;19:1099-111.
<https://doi.org/10.1109/TIFS.2023.3331239>.

Chundawat VS, Tarun AK, Mandal M, Kankanhalli M. Zero-shot machine unlearning. *IEEE Trans Inf Forensics Secur*. 2023;18:2345-54.
<https://doi.org/10.1109/TIFS.2023.3265506>.

Tarun AK, Chundawat VS, Mandal M, Kankanhalli M. Fast yet effective machine unlearning. *IEEE Trans Neural Netw Learn Syst*. 2024;35(9):13046-55.
<https://doi.org/10.1109/TNNLS.2023.3266233>.

Abbasi A, Thrash C, Akbari E, Zhang D, Kolouri S. CovarNav: machine unlearning via model inversion and covariance navigation. *arXiv [Preprint]*. 2023;arXiv:2311.12999.

Yoon Y, Nam J, Yun H, Lee J, Kim D, Ok J. Few-shot unlearning by model inversion. *arXiv [Preprint]*. 2022;arXiv:2205.15567.

Ma Z, Liu Y, Liu X, Liu J, Ma J, Ren K. Learn to forget: machine unlearning via neuron masking. *IEEE Trans Dependable Secur Comput*. 2023;20(4):3194-207.
<https://doi.org/10.1109/TDSC.2022.3194884>.

De Lange M, Aljundi R, Masana M, Parisot S, Jia X, Leonardis A, et al. A continual learning survey: defying forgetting in classification tasks. *IEEE Trans Pattern Anal Mach Intell*. 2022;44(7):3366-85.

<https://doi.org/10.1109/TPAMI.2021.3057446>.

Wang L, Zhang X, Su H, Zhu J. A comprehensive survey of continual learning: theory, method and application. *IEEE Trans Pattern Anal Mach Intell*. 2024;46(8):5362-83.

<https://doi.org/10.1109/TPAMI.2024.3367329>.

Parisi GI, Kemker R, Part JL, Kanan C, Wermter S. Continual lifelong learning with neural networks: a review. *Neural Netw*. 2019;113:54-71.

<https://doi.org/10.1016/j.neunet.2019.01.012>.

Masana M, Liu X, Twardowski B, Menta M, Bagdanov AD, van de Weijer J. Class-incremental learning: survey and performance evaluation on image classification. *IEEE Trans Pattern Anal Mach Intell*. 2023;45(5):5513-33.

<https://doi.org/10.1109/TPAMI.2022.3213473>.

Kong Y, Liu L, Chen H, Kacprzyk J, Tao D. Overcoming catastrophic forgetting in continual learning by exploring eigenvalues of Hessian matrix. *IEEE Trans Neural Netw Learn Syst*. 2024;35:16196-210.

Peng J, Tang B, Jiang H, Li Z, Lei Y, Lin T, et al. Overcoming long-term catastrophic forgetting through adversarial neural pruning and synaptic consolidation. *IEEE Trans Neural Netw Learn Syst*. 2022;33:4243-56.

Zhang M, Li H, Pan S, Chang X, Zhou C, Ge Z, et al. One-shot neural architecture search: maximising diversity to overcome catastrophic forgetting. *IEEE Trans Pattern Anal Mach Intell*. 2021;43(9):2921-35.

<https://doi.org/10.1109/TPAMI.2020.3035351>.

French RM. Catastrophic forgetting in connectionist networks. *Trends Cogn Sci*. 1999;3(4):128-35.

[https://doi.org/10.1016/S1364-6613\(99\)01294-2](https://doi.org/10.1016/S1364-6613(99)01294-2).

LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. *Proc IEEE*. 1998;86(11):2278-324.

<https://doi.org/10.1109/5.726791>.

Xiao H, Rasul K, Vollgraf R. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv [Preprint]*. 2017;arXiv:1708.07747.

Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. *Adv Neural Inf Process Syst*. 2012;25:1097-105.

Netzer Y, Wang T, Coates A, Bissacco A, Wu B, Ng AY. Reading digits in natural images with unsupervised feature learning. In: *Proceedings of the NIPS Workshop on Deep Learning and Unsupervised Feature Learning*; 2011; Granada, Spain.

Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. *arXiv [Preprint]*. 2014;arXiv:1409.1556.

He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*; 2016; Las Vegas, NV. p.770-8.

<https://doi.org/10.1109/CVPR.2016.90>.

Krizhevsky A, Hinton G. Learning multiple layers of features from tiny images [technical report]. Toronto: University of Toronto; 2009.

Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z. Rethinking the Inception architecture for computer vision. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*; 2016; Las Vegas, NV. p.2818-26.

<https://doi.org/10.1109/CVPR.2016.308>.

Lee SH, Lee S, Song BC. Vision transformer for small-size datasets. *arXiv [Preprint]*. 2021;arXiv:2112.13492.

Naqvi SAR. Augmented skin conditions image dataset [Internet]. Kaggle; 2023 [cited 2024 Oct 31]. Available from: Kaggle.

Bourtole L, Chandrasekaran V, Choquette-Choo CA, Jia H, Travers A, Zhang B, et al. Machine unlearning. In: Proceedings of the 2021 IEEE Symposium on Security and Privacy (SP); 2021; San Francisco, CA. Piscataway (NJ): IEEE; 2021. p.141-59.
<https://doi.org/10.1109/SP40001.2021.00019>.