computers

MDPI

Systematic Review

A Systematic Literature Review of Machine Unlearning Techniques in Neural Networks

Ivanna Daniela Cevallos †, Marco E. Benalcázar *,†, Ángel Leonardo Valdivieso Caraguay †, Jonathan A. Zea † and Lorena Isabel Barona-López †

Artificial Intelligence and Computer Vision Research Laboratory "Alan Turing", Departamento de Informática y Ciencias de la Computación (DICC), Escuela Politécnica Nacional, Quito 170525, Ecuador; ivanna.cevallos@epn.edu.ec (I.D.C.); angel.valdivieso@epn.edu.ec (Á.L.V.C.); jonathan.zea@epn.edu.ec (J.A.Z.); lorena.barona@epn.edu.ec (L.I.B.-L.)

* Correspondence: marco.benalcazar@epn.edu.ec

† These authors contributed equally to this work.

Abstract: This review examines the field of machine unlearning in neural networks, an area driven by data privacy regulations such as the General Data Protection Regulation and the California Consumer Privacy Act. By analyzing 37 primary studies of machine unlearning applied to neural networks in both regression and classification tasks, this review thoroughly evaluates the foundational principles, key performance metrics, and methodologies used to assess these techniques. Special attention is given to recent advancements up to December 2023, including emerging approaches and frameworks. By categorizing and detailing these unlearning techniques, this work offers deeper insights into their evolution, effectiveness, efficiency, and broader applicability, thus providing a solid foundation for future research, development, and practical implementations in the realm of data privacy, model management, and compliance with evolving legal standards. Additionally, this review addresses the challenges of selectively removing data contributions at both the client and instance levels, highlighting the balance between computational costs and privacy guarantees.

Keywords: machine unlearning; data privacy; neural networks; selective forgetting

check for updates

Academic Editors: Jeremy Straub, Hussain Mohammed Dipu Kabir, Syed Bahauddin Alam and Subrota Kumar Mondal

Received: 28 January 2025
Revised: 28 March 2025
Accepted: 29 March 2025
Published: 18 April 2025

Citation: Cevallos, I.D.; Benalcázar, M.E.; Valdivieso Caraguay, Á.L.; Zea, J.A.; Barona-López, L.I. A Systematic Literature Review of Machine Unlearning Techniques in Neural Networks. *Computers* **2025**, *14*, 150. https://doi.org/10.3390/computers14040150

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

In response to data privacy regulations like the General Data Protection Regulation (GDPR) and California Consumer Privacy Act (CCPA), the concept of the 'Right to be Forgotten' has gained visibility. These regulations impose compliance burdens on organizations by requiring them to implement mechanisms for data erasure. Specifically, the GDPR stipulates that individuals have the right to demand the deletion of their data if it is no longer necessary for its original purpose or if they withdraw consent for its processing [1]. Moreover, these deletions must be performed promptly, within a timeframe known as "without undue delay" [2]. It is increasingly recognized that data deletion should not be limited to databases but should extend to the removal of personal data from machine learning models themselves. For instance, a data regulator in the United Kingdom has warned businesses about machine learning software falling under GDPR provisions. Similarly, the US Federal Trade Commission had required Paravision, a facial recognition startup, to erase a collection of facial images. These images were improperly acquired. They also had to erase the machine learning models trained using these images [3]. Machine unlearning emerges as an area of research in response to these regulatory mandates. It refers to

modifying trained machine learning models to selectively forget specific subsets of data, thereby ensuring compliance with deletion requests without the need for complete model retraining [4]. Retraining becomes expensive. Studies found that retraining large machine learning models like GPT-3 can cost hundreds of thousands of dollars in computational resources alone [5]. Consequently, the ability to unlearn becomes essential not only for protecting individuals' privacy but also for mitigating legal and financial risks associated with non-compliance.

1.1. Nomenclature

Table 1 provides a detailed breakdown of the nomenclature used throughout this document for clarity and reference. The nomenclature serves as a guide to understanding the various elements and entities involved in machine learning and machine unlearning processes.

Table 1. Common nomenclature used throughout the review related to machine unlearning.

Symbol	Description
x	Input data sample
y	Predicted output
x_u	Data point to be unlearned
D	Entire dataset
D_u	Subset of dataset to be unlearned
D_r	Remaining dataset after unlearning
L	Loss function
α	Learning rate
θ	Parameters
∇L	Gradient of the loss function
H	Hypothesis space
F	Feature space
G	Task space
W	Weights
b	Bias vector
ℓ	Layer
z	Logits or preactivation values
$\mathcal{A}$	Training algorithm
M	Machine learning model trained on D
N	Number of samples in the dataset
$\mathcal{N}$	Noise matrix
U	Unlearning process
P_θ	Distribution of model parameters
K	Similarity measure
M'	Unlearned model

1.2. Overview and Contributions

While existing reviews on machine unlearning provide summaries of methodologies and taxonomies of techniques, there remain significant gaps in understanding their limitations and areas needing further investigation. Specifically, current techniques face

unresolved challenges regarding scalability, the precision of unlearning at different levels (class-level versus individual data points), and reproducibility across various architectures and datasets. Moreover, the effectiveness of these techniques in neural networks, particularly in the contexts of regression and classification tasks, has not been thoroughly assessed.

This survey addresses these gaps by focusing exclusively on machine unlearning techniques applied to neural networks and categorizing them based on foundational principles and mathematical frameworks. It provides a chronological ordering of techniques up to December 2023, analyzing their evolution, interrelationships, and contributions towards overcoming existing limitations. Additionally, this work evaluates the reproducibility of these techniques across different experimental setups, compares datasets and model architectures, and analyzes the achieved levels of unlearning. The aim is to provide a clearer understanding of the strengths and weaknesses of existing approaches, as well as to identify areas requiring further exploration.

By presenting this systematic review, the survey not only categorizes existing techniques but also serves to highlight current challenges and provide insights for advancing the field of machine unlearning in neural networks.

1.3. General Objective

Conduct a systematic literature review on machine unlearning in neural networks for classification and regression tasks.

1.4. Specific Objectives

1. Define the categorization framework for machine unlearning techniques applicable to neural networks in regression and classification contexts.
2. Identify the foundational principles underlying different machine unlearning techniques.
3. Analyze the metrics and methodologies commonly used to assess the efficacy of machine unlearning techniques.

The structure of this document is designed to address the aspects of machine unlearning. Section 1 provides an overview of the research topic, its significance, and the contribution of the study. Section 2 outlines the research questions and the process of selecting primary studies. It includes a description of the research methodology used to gather and analyze data. Section 3 provides the nomenclature used and the definitions of key terms and concepts related to machine unlearning, discusses the practical applications of machine unlearning, and explores the challenges and obstacles encountered in its implementation. Section 4 examines the challenges and obstacles faced in the implementation of machine unlearning techniques. Section 5 presents the practical applications of machine unlearning techniques in various domains, highlighting their significance and relevance. Section 6 is divided into subsections that categorize and examine different unlearning techniques based on databases, architecture, and federated learning. This section provides a detailed analysis of the methodologies used in each category. Section 7 interprets the findings of the study. Finally, Section 8 summarizes the key findings, draws conclusions, and suggests directions for future research.

2. Methodology

This review was performed in accordance with the PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) guidelines. It also follows Kitchenham's methodology [6]. First, specific research questions were defined to guide the scope and focus of the review. Second, a comprehensive search of primary studies was conducted from relevant academic databases and sources to gather pertinent literature. Third, the

primary studies were analyzed by examining the collected literature for quality, relevance, and contributions to the field. Fourth, essential information and findings were extracted from the studies.

2.1. Research Questions

This review aims to categorize and evaluate various machine unlearning techniques within neural network models; consequently, it is guided by three research questions:

1. RQ1: How can machine unlearning techniques, for neural networks with regression or classification tasks, be categorized?
2. RQ2: What are the foundational principles underlying different machine unlearning techniques?
3. RQ3: What metrics and methods are commonly used to evaluate the effectiveness of machine unlearning techniques in different datasets and architectural setups?

2.2. Search for Primary Studies

The process begins by selecting relevant academic databases and repositories. Five key sources were chosen for their comprehensive coverage of machine learning literature: ACM, IEEE, Science Direct, Springer, and ArXiv. The first four databases were selected due to their extensive collection of peer-reviewed primary studies, which ensures the inclusion of high-quality, validated research. ArXiv was included despite the lack of peer reviews because it provides timely access to the latest insights and developments, which is essential in the rapidly evolving field of machine learning. Prioritizing ArXiv allows the survey to capture emerging techniques and trends that may not yet be covered in peer-reviewed venues, ensuring a more up-to-date perspective on the topic.

The next step involves identifying keywords related to the research questions to create search queries. The chosen keywords include machine unlearning, forgetting, mechanism, data, removal, neural network, classification, regression, and federated unlearning. Using these keywords, Search Strings (SS) were developed with Boolean operators "AND" and "NOT" to refine the queries. The terms "generative" and "catastrophic" were explicitly excluded from the search. The exclusion of "generative" aligns with the focus on specific neural network architectures, as outlined in the Introduction, which does not encompass generative models. "Catastrophic forgetting" was excluded because it refers to the unintended loss of previously learned information when a neural network is trained on new data, whereas this survey concentrates on techniques that allow for the controlled and selective modification or removal of previously learned information.

These criteria and search strategies were established to ensure the inclusion of studies most relevant to the research objectives. By using a combination of peer-reviewed and preprint sources, the survey aims to balance the rigor of established research with the innovative findings presented in preprints. The detailed list of Search Strings and its corresponding ID SS_i can be found in Table 2.

As mentioned before, this review on machine unlearning in neural networks will focus on studies published from January 2015 to December 2023. This timeframe was chosen because the first relevant paper in the field appeared in 2015. Additionally, papers published after this period have frequently cited this earlier work. Extending the review to December 2023 aims to capture the most recent advancements and discussions in this rapidly evolving area, ensuring a comprehensive and up-to-date analysis. Figure 1 illustrates the distribution of these studies per year, offering a visual representation of the research trend in this domain.

Table 2. Search Strings (SS_i) used to find primary studies.

ID	Search String
SS1	"machine unlearning" AND "neural network" AND ("CLASSIFICATION" OR "REGRESSION") NOT "generative" NOT "catastrophic"
SS2	"machine forgetting" AND "neural network" AND ("CLASSIFICATION" OR "REGRESSION") NOT "generative" NOT "catastrophic"
SS3	"forgetting mechanism" AND "neural network" AND ("CLASSIFICATION" OR "REGRESSION") NOT "generative" NOT "catastrophic"
SS4	"algorithmic forgetting" AND "neural network" AND ("CLASSIFICATION" OR "REGRESSION") NOT "generative" NOT "catastrophic"
SS5	"Data Removal" AND "neural network" AND ("CLASSIFICATION" OR "REGRESSION") NOT "generative" NOT "catastrophic"

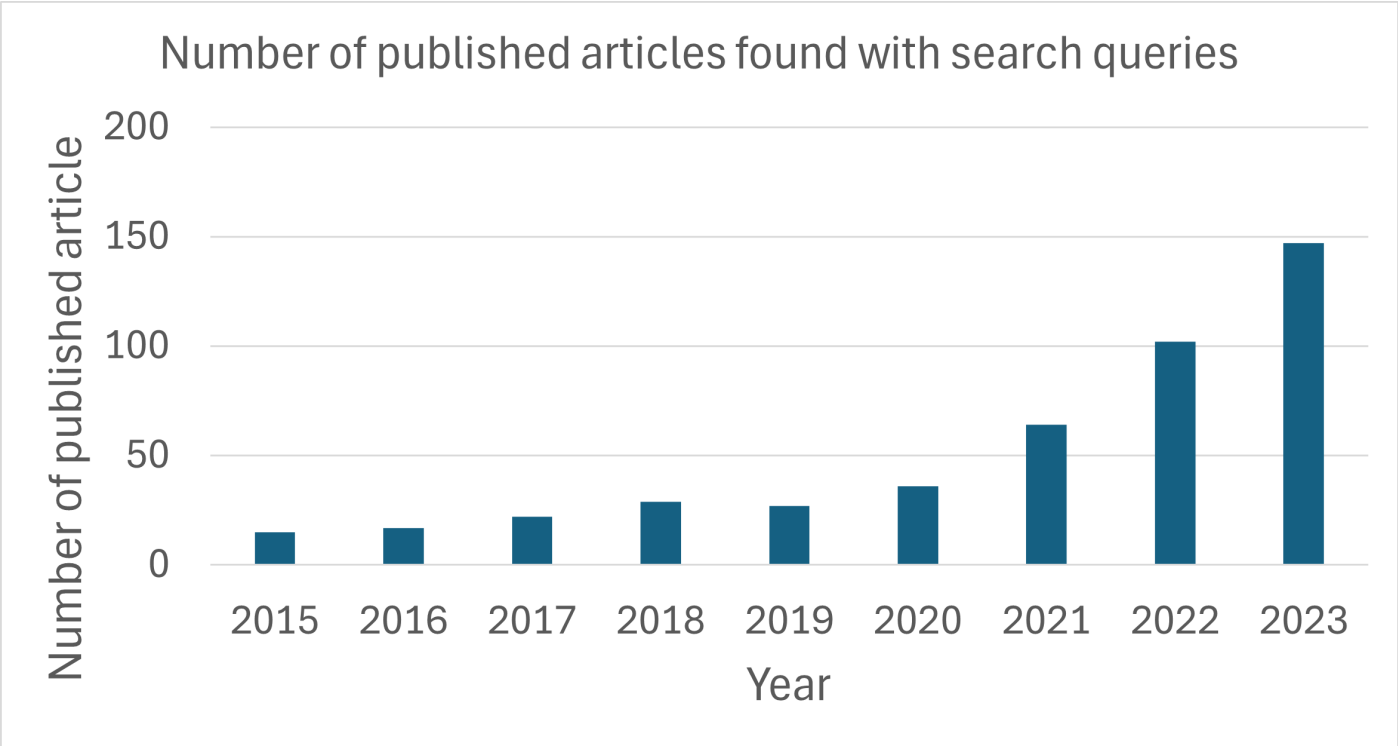

Figure 1. Annual distribution of studies on machine unlearning in neural networks.

There were 445 primary studies found in the four repositories. Then, 118 duplicate studies were removed and 11 primary studies were added using the snowballing techniques. As a result, a total of 348 primary studies were identified. The number of included and excluded papers for each phase is presented in Figure 2 using the PRISMA flowchart.

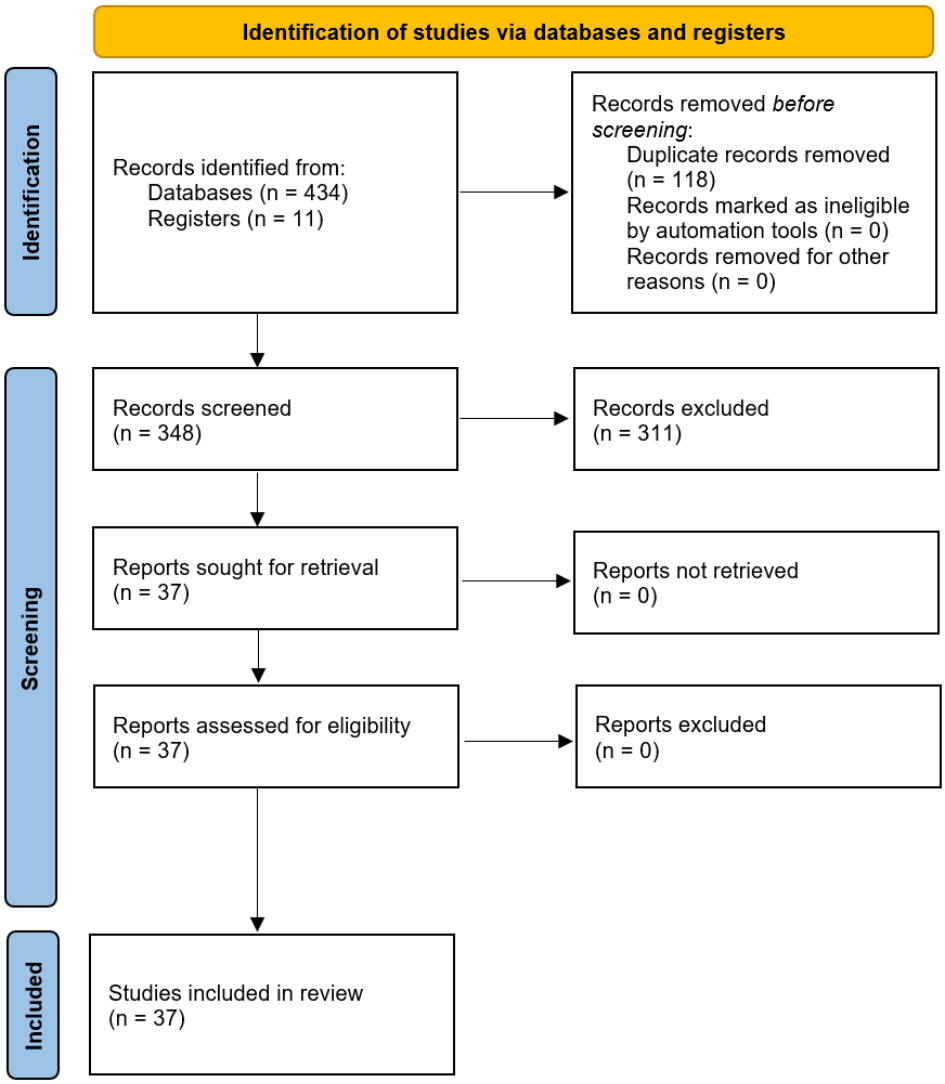

Figure 2. Systematic review flowchart.

2.3. Analysis of Primary Results

The 348 primary studies were filtered by evaluating the titles, abstracts, and conclusions according to the inclusion and exclusion criteria and the assessment questions. One inclusion criterion stated that the studies must be accessible in full text for a comprehensive review, and research from preprint servers was included to capture the latest developments in the field. Studies that present frameworks or methodologies designed for unlearning in machine learning were also included. Specific exclusion criteria were applied, such as omitting conference abstracts, editorials, and opinion pieces without empirical research data or detailed methodologies. Additionally, studies that were not published in English and studies focused on methods related to catastrophic forgetting or unintentional data forgetting were excluded. Studies were also excluded if the models were not neural networks or if they were not used for regression and classification tasks. Excluding generative models and the concept of catastrophic forgetting aimed to maintain focus on deliberate unlearning mechanisms. This selection process ultimately narrowed down the focus to 37 primary studies, as shown in Table 3, providing a foundation for the review. It gives a clear view of the title of the paper, its identifier, and the name of the proposed technique or a short name of the title if no name for the technique was given in the paper. This will facilitate the organization and retrieval of relevant information.

Table 3. Primary Studies.

ID	Title	Authors	Year	Technique
1	Trojaning Attack on Neural Networks [7]	Liu et al.	2018	BadNets [2]
2	Class Clown: Data Redaction in Machine Unlearning at Enterprise Scale [8]	Felps et al.	2020	Class Clown [2]
3	Fast Yet Effective Machine Unlearning [9]	Tarun et al.	2024	Fast yet effective machine unlearning [2]
4	Learning with Selective Forgetting [10]	Shibata et al.	2021	Mnemonic code [2]
5	Machine Unlearning [11]	Bourtoule et al.	2020	SISA [1]
6	Adaptive Machine Unlearning [12]	Gupta et al.	2021	Adaptive Machine Unlearning [2]
7	No Matter How You Slice It: Machine Unlearning with SISA Comes at the Expense of Minority Classes [13]	Koch et al.	2023	No matter how you slice it [2]
8	Coded Machine Unlearning [14]	Aldaghri et al.	2021	Coded machine unlearning [1]
9	DeepObliviate: A Powerful Charm for Erasing Data Residual Memory in Deep Neural Network [15]	He et al.	2021	DeepObliviate [2]
10	ARCANE: An Efficient Architecture for Exact Machine Unlearning [16]	Yan et al.	2022	ARCANE [2]
11	Amnesiac Machine Learning [17]	Graves et al.	2020	Amnesiac Machine Unlearning [2]
12	Unrolling SGD: Understanding Factors Influencing Machine Unlearning [18]	Thudi et al.	2022	Unrolling SGD [1]
13	Learn to Forget: Machine Unlearning via Neuron Masking [19]	Liu et al.	2021	Forsaken [2]
14	Backdoor Defense with Machine Unlearning [20]	Liu et al.	2022	BAERASER [2]
15	Can Bad Teaching Induce Forgetting? [21]	Chundawat et al.	2023	Bad teaching [2]
16	Zero-Shot Machine Unlearning [22]	Chundawat et al.	2023	Gated Knowledge Transfer [2]
17	Efficient Two-stage Model Retraining for Machine Unlearning [23]	Kim et al.	2022	Efficient two-stage model [2]
18	Towards Unbounded Machine Unlearning [24]	Kurmanji et al.	2023	Towards Unbounded Machine Unlearning [2]
19	Lightweight machine unlearning in neural network [25]	Chen et al.	2021	Lightweight machine unlearning [2]
20	Deep Regression Unlearning [26]	Tarun et al.	2023	Deep Regression Unlearning [2]
21	Eternal Sunshine of the Spotless Net: Selective Forgetting in Deep Networks [27]	Golatkar et al.	2020	Eternal Sunshine [2]
22	Forgetting Outside the Box: Scrubbing Deep Networks of Information Accessible from Input-Output Observation [28]	Golatkar et al.	2020	Forgetting outside the box [2]
23	Mixed-Privacy Forgetting in Deep Networks [29]	Golatkar et al.	2021	Mixed privacy [2]

Table 3. *Cont.*

ID	Title	Authors	Year	Technique
24	Certified Data Removal from Machine Learning Models [30]	Guo et al.	2023	Certified Removal [2]
25	Approximate Data Deletion from Machine Learning Models [31]	Izzo et al.	2021	Projective Residual Update [2]
26	PUMA:Performance Unchanged Model Augmentation for Training Data Removal [32]	Wu et al.	2022	Performance Unchanged Model Augmentation [2]
27	Machine Unlearning of Features and Labels [33]	Warnecke et al.	2023	Unlearn Features and labels [2]
28	FedEraser: Enabling Efficient Client-Level Data Removal from Federated Learning Model [34]	Liu et al.	2021	FedEraser [2]
29	Federated Unlearning with Knowledge Distillation [35]	Wu et al.	2022	FU with Knowledge distillation [2]
30	Federated Unlearning via Class-Discriminative Pruning [36]	Wang et al.	2022	Class-Discriminative Pruning [2]
31	The Right to be Forgotten in Federated Learning: An Efficient Realization with Rapid Retraining [37]	Liu et al.	2022	Efficient Realization with Rapid Retraining [2]
32	FedRecover: Recovering from Poisoning Attacks in Federated Learning using Historical Information [38]	Cao et al.	2022	FedRecover [2]
33	Subspace based Federated Unlearning [39]	Li et al.	2023	Subspace-based Unlearning [2]
34	Asynchronous Federated Unlearning [40]	Su et al.	2023	Asynchronous Federated Unlearning [2]
35	Heterogeneous Decentralized Machine Unlearning with Seed Model Distillation [41]	Ye et al.	2023	Seed Model Distillation [2]
36	Federated Unlearning: How to Efficiently Erase a Client in FL? [42]	Halimi et al.	2023	Federated Client-Level Unlearning [2]
37	QuickDrop: Efficient Federated Unlearning by Integrated Dataset Distillation [43]	Dhasade et al.	2023	Integrated Dataset Distillation [2]

[1] Type of unlearning: exact unlearning. [2] Type of unlearning: approximate unlearning.

3. Theoretical Framework

This section delves into key concepts essential for understanding machine unlearning techniques, explores diverse scenarios where these techniques are applied, and examines the obstacles encountered in their implementation.

3.1. Definitions

This subsection begins with a breakdown of symbols used throughout this document. Additionally, definitions are provided for concepts such as machine unlearning, exact machine unlearning, and approximate machine unlearning, with capture complex setting the stage for understanding.

3.1.1. Machine Unlearning

The term "machine unlearning" was introduced in "Towards Making Systems Forget with Machine Unlearning" [44]. The authors of [44] proposed an unlearning algorithm that reformulates the learning process into a summation format. By updating only a small part of these summations, their method achieves significantly faster unlearning compared to retraining the model from scratch. However, this approach is limited to traditional machine learning techniques that can be represented in a summation form. Building on this foundational idea, machine unlearning has come to refer more broadly to the process in which the influence of a specific subset of a training dataset, denoted as D_u, is removed from a trained machine learning model. The goal is to modify the model M so that it performs as if it had never encountered the data in D_u. This is achieved through an unlearning process, $U(A(D), D, D_u)$, which adjusts the model's parameters or data to exclude the effects of the unlearning dataset D_u while retaining the influence of the remaining dataset D_r.

3.1.2. Exact Machine Unlearning

Exact unlearning ensures that the modified machine learning model behaves as though it never encountered the unlearned data subset. This means that after the unlearning process, the model's predictions, outputs, and behaviors will be statistically identical to those produced by a model that was retrained from scratch using the remaining dataset, excluding the subset of unlearned data [27]. The goal is to ensure that no identifiable impact or knowledge of the unlearned data influences the model's performance or outputs, maintaining the integrity and confidentiality of the training process. However, one of the drawbacks of exact unlearning is its limited applicability. Complex models, due to their intricate architectures and numerous parameters, may not allow for such an exact replication of the model's original state after unlearning. This limitation may necessitate alternative approaches, such as approximate unlearning, which might allow for minor deviations in behavior.

3.1.3. Approximate Machine Unlearning

This unlearning method ensures that the modified model and a model retrained from scratch are approximately indistinguishable in their output. Typically, this approximation is achieved using differential privacy techniques, such as ϵ-δ certified unlearning [30]. In this context, the ϵ-δ certified unlearning approach bounds the divergence between the output of the unlearned model and the retrained model to a defined threshold. Specifically, ϵ-δ certified unlearning ensures that the divergence between the two models remains within a tolerable margin.

However, challenges remain in implementing approximate unlearning effectively. For instance, the degree of privacy and tolerance levels can affect the model's accuracy and performance, and ensuring that this balance does not compromise the quality of the unlearned model is essential. Additionally, different model architectures and loss functions may impact the efficacy and efficiency of the unlearning process, making it imperative to tailor these techniques to specific scenarios [45].

Many methods aim for "exact" unlearning by achieving statistical indistinguishability from a model trained without forgotten data [26]. However, achieving statistical indistinguishability does not necessarily mean that the models are mathematically equivalent or that all traces of the data are completely removed [16]. Techniques that strive for exact unlearning often rely on approximations and assumptions that limit their ability to achieve true equivalence with a model trained from scratch [24]. The stochastic nature of training algorithms introduces variability in model parameters [11], making it difficult to achieve a truly exact method in practice. In federated learning, the lack of access to client datasets and

the stochastic nature of training pose additional challenges, making it difficult to achieve exact unlearning. Any approximation method is likely to fail to reach the same result as training without the unlearned data [35].

3.1.4. Differential Privacy

Differential privacy (DP) (Figure 3) is a foundational framework in data privacy that ensures individuals are not adversely affected by allowing their data to be used in studies or analyses [46]. It provides a promise by data curators to data subjects that their participation in data analysis will not result in any negative consequences, regardless of the availability of other datasets or information sources. While DP can naturally achieve machine unlearning by ensuring that the presence of a sample in the training data cannot be discerned from the model, it primarily focuses on protecting the privacy of all samples to some extent. DP imposes a subtle bound on the contribution of each sample to the final model, but it cannot completely constrain the contribution to zero without rendering the model ineffective for learning from the training data. On the other hand, machine unlearning aims to completely cancel the contribution of a target sample, effectively removing its influence from the model. Consequently, machine unlearning (MU) and DP operate on different principles, with MU seeking to eliminate specific data contributions entirely.

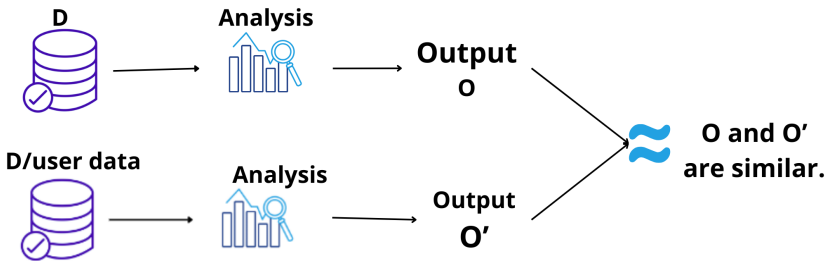

Figure 3. Scheme of differential privacy applied to machine learning models.

3.1.5. Federated Learning

Federated learning (Figure 4) operates as a decentralized method in machine learning, where numerous clients participate in training a global model without sharing their raw data [47]. Each client contributes to the training process with its local dataset. The global model's parameters are updated collaboratively across all clients through iterative rounds of communication and computation, where each client computes model updates based on its local data and transmits them to a central server. The central server aggregates these updates to refine parameters, aiming to improve the global model's performance while preserving the privacy of individual datasets.

3.2. Metrics

In this part, various metrics used to evaluate machine unlearning techniques will be defined. The lack of a uniform metric for unlearning stems from the diverse goals, definitions, and evaluation methods employed by different approaches. Each unlearning technique aims to address a specific facet of the problem, which leads to the use of varied metrics. This makes direct comparisons challenging and highlights the need for a more unified framework to evaluate unlearning methods. There are two methods to assess a machine unlearning technique: evaluation metrics and verification methods. Evaluation metrics serve as theoretical criteria for assessing unlearning efficacy; for example, accuracy on forget set or retain set, error rate, relearn time, Anammesis Index, and distance metric are used for this purpose. Verification methods aim to ensure that one cannot easily distinguish between unlearned models and their retrained counterparts. Some examples of verification methods are attacks and unlearning cost. In Section 6, "Analysis of Techniques", the

performance of each technique on these metrics will be presented, along with comparisons to other baselines and techniques.

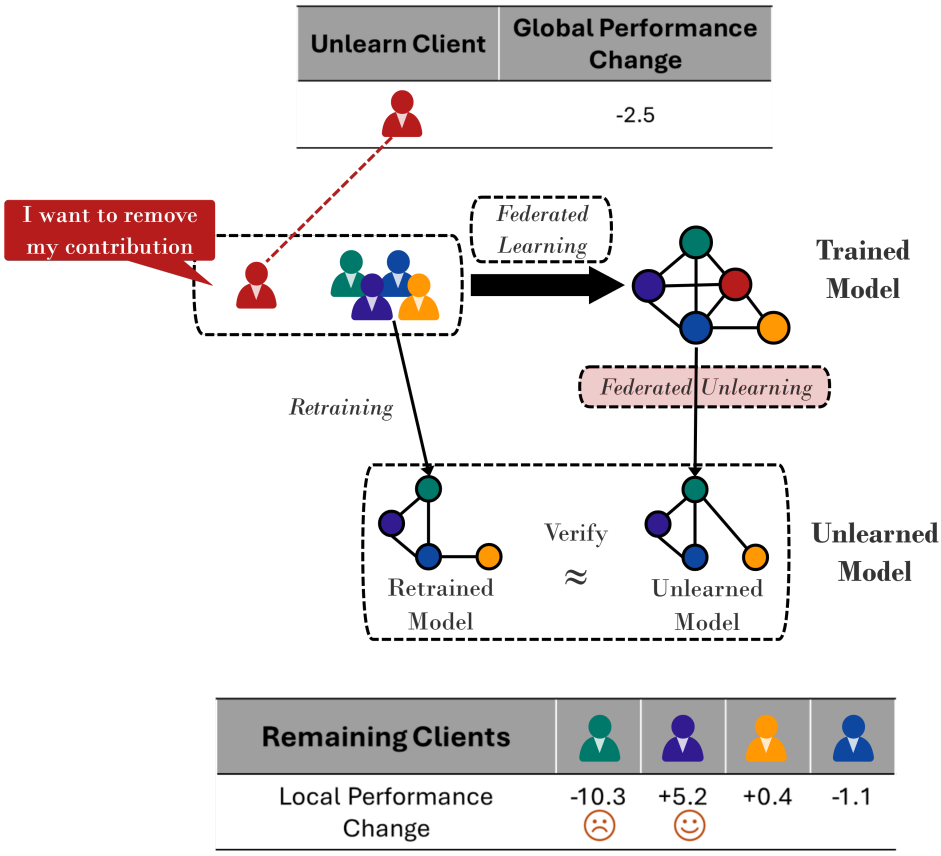

Figure 4. Representation of federated learning and federated unlearning. Source adapted from [48].

3.2.1. Accuracy on Forget Set

Accuracy measures the proportion of correctly classified instances out of the total instances in a dataset [49]. It is calculated as the number of correct predictions divided by the total number of predictions, often expressed as a percentage. In machine unlearning techniques, the accuracy is measured on the forget set D_u and refers to the model's performance on the subset of data designated for unlearning. The goal of accuracy on D_u is to be close to that of the retrained model. Ideally, this accuracy should be low [22].

3.2.2. Accuracy on Retain Set

Accuracy on the retain set D_r refers to the model's performance on the data subset that remains unchanged after unlearning. The goal of accuracy on D_r is to closely match the performance of the original model before unlearning. This metric assesses how well the model retains its classification capability on the data it was initially trained on [22].

3.2.3. Error Rate

Error Rate is calculated as $1 -$ Accuracy on retain set. It measures the proportion of misclassified instances in the data subset that remains unchanged after unlearning [13].

3.2.4. Relearn Time

Relearn time measures the model's retention of information about the unlearned data. It acts as a way to measure how quickly the model can recover its performance on the unlearned data by retraining. If the model achieves comparable performance to the source

model with minimal retraining epochs, it suggests that residual information about the unlearned data persists within the model [9].

3.2.5. Anamnesis Index

The Anamnesis Index evaluates how quickly the unlearned model can regain a certain level of accuracy by comparing its relearn time with that of a model trained from scratch on the retained data. It normalizes the relearn time by considering a margin of α% around the model's original accuracy before unlearning. This metric not only measures how quickly the model can relearn but also assesses the effectiveness of the unlearning process [22].

3.2.6. Distance

Another way to evaluate the effectiveness of an approximate data deletion method is by measuring the ℓ_2 distance between the estimated model parameters and those obtained through complete retraining. When the parameters from the unlearning model closely align with the model fully retrained, it indicates that both models are likely to make similar predictions [31].

3.2.7. Attacks

The metric evaluates the success of unlearning models based on their ability to mitigate membership inference attacks and backdoor infection scenarios. These studies involve simulations where adversaries attempt to infiltrate and compromise the model's privacy and integrity. In the next section, further details on these evaluations will be discussed in depth.

3.2.8. Unlearning Cost (Storage and Time Cost)

This refers to the resources, both in terms of storage capacity and computational time, required to implement the unlearning process effectively. The unlearning cost includes the storage space needed to maintain original model parameters, intermediate states during unlearning, and redundant data. It also encompasses the time taken to execute the unlearning procedure, which involves iterative processes to remove or adjust trained data.

4. Challenges

Machine unlearning faces challenges from both the inherent properties of machine learning models and practical implementation issues.

4.1. Stochastic

The stochastic nature of training in modern machine learning pipelines introduces complexities that hinder effective unlearning strategies [4]. This stochasticity arises from various factors, including the random sampling of small batches from the dataset during training, the unpredictable ordering of batches across epochs, and the parallelization of training without explicit synchronization, leading to non-deterministic behavior. Furthermore, training is an incremental process where updates depend on prior updates, amplifying the impact of stochasticity throughout the learning procedure. This incremental nature, coupled with the inherent randomness in learning algorithms such as stochastic gradient descent, poses significant challenges in understanding how individual data points influence the learned model.

4.2. Streisand Effect

The misuse of scrubbing procedures can inadvertently amplify the visibility of forgotten information—a phenomenon known as the "Streisand effect" [50]. Originating from Barbara Streisand's attempt to restrict online access to her residence, the term refers to

the unintended consequence of heightened attention resulting from efforts to suppress information.

4.3. Data Interconnections

Machine learning models do not simply analyze individual data points in isolation. Instead, they collaboratively extract intricate statistical patterns and interdependencies among data points [51]. These interconnections can be deeply embedded within the model's learned representations. Removing a single data point can disrupt these learned patterns and interconnections, potentially causing a notable performance decline. This complexity underscores a significant challenge in machine unlearning: effectively removing the influence of specific data points while preserving the model's overall performance and coherence. The challenge lies in disentangling the contributions of individual data points from the interconnected structure of the model without negatively impacting its ability to generalize from the remaining data.

4.4. Uncertainty

Machine unlearning has become a crucial area of research due to increasing privacy regulations and security concerns. However, selectively removing knowledge from neural networks introduces various types of uncertainty that can impact model stability, generalization, and reliability [52]. The main sources of uncertainty in machine unlearning and the recent approaches to mitigate their effects are as follows:

- Epistemic uncertainty arises from the model's limited knowledge about the underlying data distribution. When specific training data points are removed, the model's uncertainty can increase, leading to instability in its predictions [53,54]. Researchers have introduced methods that measures deviations in model parameters after unlearning compared to a fully retrained model [54].
- Aleatoric uncertainty is related to inherent randomness in the data. The process of unlearning can modify the variance of model parameters, affecting prediction consistency. Recent studies highlight that some unlearning methods based on gradient ascent can amplify aleatoric uncertainty unless suitable regularization techniques are applied [55].

5. Application

In addition to ensuring compliance with data protection regulations, as discussed in the introduction, machine unlearning offers a wide range of applications and benefits. This section will explore these broader applications, demonstrating how machine unlearning techniques can address various challenges and improve modern machine learning practices.

5.1. Prevent Backdoor Injection Attack

A backdoor injection attack is a malicious manipulation of a machine learning model's behavior, where an adversary strategically implants a trigger pattern into the training data to induce the model to exhibit specific, undesired behaviors upon encountering inputs containing the trigger pattern [20]. The attacker aims to modify the model's decision boundary such that inputs augmented with the trigger pattern are classified into a targeted label, regardless of their original labels. This attack manipulates the model's predictions, leading to a compromised system vulnerable to adversarial manipulation. Machine unlearning aids in backdoor defense by strategically eliminating the influence of specific trigger patterns introduced by attackers on the victim model. It achieves this by reversing the backdoor injection process and erasing the memorized trigger patterns from the model's learned representations.

5.2. Prevent Membership Inference Attacks

Membership inference attacks aim to determine whether a specific data point was part of the training data for a machine learning model [56]. This attack exploits the inadvertent leakage of information contained within a model's outputs, enabling adversaries to infer the presence or absence of individual data points in the training dataset. Machine unlearning techniques, designed to remove or mitigate the influence of certain data points on a model's parameters, can serve as a defense mechanism against membership inference attacks. By systematically eliminating the association between sensitive data points and the model's parameters, unlearning disrupts the adversary's ability to infer membership status accurately.

5.3. Fast Model Debias

Bias in machine learning models arises from systematic errors or prejudices in predictions, often stemming from skewed or incomplete training data. These biases can lead to unfair outcomes, perpetuating social disparities and undermining prediction reliability. To mitigate this issue, [57] advocates for the application of machine unlearning techniques as a debiasing tool. Unlike previous methods that often require costly human labeling or computationally intensive model retraining, machine unlearning offers a more scalable solution. The process involves first identifying the most influential harmful samples, followed by the application of machine unlearning to effectively remove associated biases. This approach addresses the limitations of traditional debiasing mechanisms and enhances fairness in models without compromising scalability or accuracy.

5.4. Enhancing Transfer Learning

Transfer learning, the process of adapting a pretrained model to a related task, often encounters challenges when the source data contain irrelevant or harmful classes for the target task. Machine unlearning techniques provide a solution by selectively removing such classes, thereby improving transfer learning accuracy. The ℓ_1-sparse MU method, proposed by [58], demonstrates significant promise in this regard. By integrating sparsity-inducing penalties into the unlearning process, this method efficiently removes undesirable data classes while preserving crucial information for the target task. The study [58] proves that ℓ_1-sparse MU achieves comparable or superior transfer learning accuracy to traditional retraining-based approaches, with the added advantage of computational efficiency, making it an appealing choice for large-scale transfer learning tasks.

5.5. Cost and Time Saving

Machine unlearning techniques offer a cost-effective alternative to traditional methods of handling personal data under regulatory frameworks like the GDPR. When data subjects invoke the "right to erasure", data controllers often face the challenging task of managing and modifying AI models to comply with regulatory requirements. Traditional solutions, such as retraining AI models using modified data sets, are time-consuming and costly. This process often involves extensive research and development costs, delays, and potential instability in AI performance, particularly when the system must relearn and adapt to the altered data environment [59]. Additionally, maintaining compliance with data privacy regulations can result in significant operational costs, especially in the EU market, where strict enforcement of privacy rules adds financial burdens not encountered in other global markets.

6. Analysis of Techniques

The taxonomy distinguishes between different approaches. An approach focuses on modifying the training data and is classified under data reorganization. Another approach involves direct adjustments to the model and is categorized as architecture-based techniques. There is also a category for federated unlearning, which focuses on client-specific data removal in decentralized models. Figure 5 summarizes the comprehensive taxonomy of machine unlearning techniques.

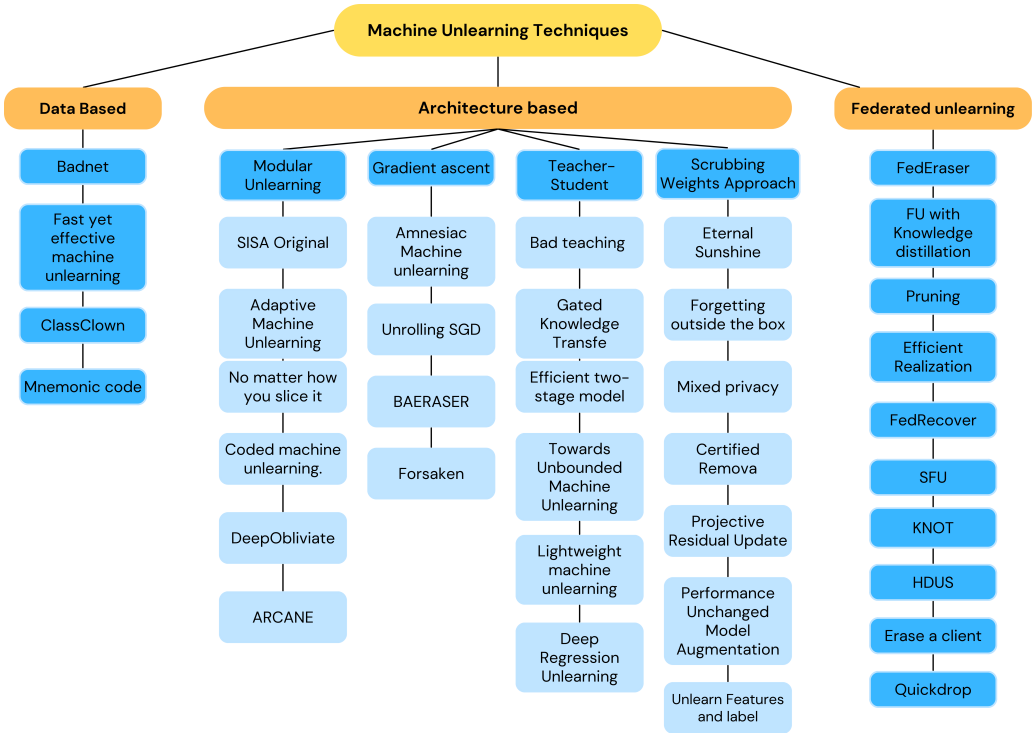

Figure 5. Taxonomy of machine unlearning techniques in neural networks.

In the rest of the section, each technique will be presented in chronological order within its corresponding category of either data-based, architecture-based, or federated unlearning. And each technique will include a definition outlining its principles and the evaluation criteria used to assess its effectiveness.

6.1. Data Based

In this section, the primary studies propose techniques that seek to unlearn via data modification. These unlearning techniques involve the alteration or crafting of specific data points to induce misclassification in machine learning models, effectively achieving unlearning.

6.1.1. BadNets

Definition

The paper [7] proposes a perspective on trojan attacks within the framework of machine unlearning, providing an approach to manipulate neural network behavior in some data point while preserving its normal functionality in the clean data. It initiates with the assumption of full access to the target neural network but lacks access to its original training or testing data. The attacker orchestrates a trojan trigger, functioning as a catalyst to induce specific misbehavior in the network. The trigger of the attack is crafted by pinpointing internal neurons strongly linked to the trigger region. The selection process for these

neurons is grounded in specific equations to quantify neuron–trigger connectivity. First, the relationship between the target layer and its preceding layer is established through:

$$\ell = \ell_{\text{preceding}} \times W + b. \tag{1}$$

To identify the most connected neurons, the following equation is used:

$$\text{argmax}_t \left(\sum_{j=0}^{n} \text{ABS}(W_{\ell(j,t)}) \right). \tag{2}$$

Here, argmax_t finds the neuron with the maximum sum of absolute weight values, $\text{ABS}(W_{\text{layer}(j,t)})$, connecting it to the preceding layer. By analyzing these weights, the neurons most strongly connected to the trigger region are identified.

This crafted trigger is then used to produce tailored training data points designed to induce misclassification in the neural network; with this misbehavior in certain data points, this technique pretends to achieve machine unlearning. The model is retrained using only these meticulously crafted data points, ensuring it operates normally under typical circumstances and misclassifies inputs targeted for unlearning.

Metric

The effectiveness of the proposed trojan attack technique is evaluated using three metrics. These include the success rate of the trojan trigger in inducing the desired misbehavior, the decrease in model accuracy on normal inputs, and the time efficiency of the attack process. The success rate is quantified by the accuracy of the trojaned model on datasets with and without the trojan trigger. The results indicate that the trojaned behavior is successfully triggered (meaning, for machine unlearning, the data point was forgotten and misclassified) in more than 92% of cases, with minimal impact on the model's performance on normal inputs (an average accuracy decrease of less than 3.5%). It demonstrates that even for complex models, trigger generation takes less than 13 min and retraining times are consistently under 4 h.

6.1.2. Class Clown

Definition

The technique [8] selectively removes sensitive data points from machine learning models without requiring full retraining. It employs intentional label poisoning during incremental retraining epochs to modify the model's behavior around identified sensitive data points. This approach aims to alter the model's decision boundaries near the redacted points, thereby reducing their susceptibility to membership inference attacks. The process utilizes stochastic gradient descent with mini-batches to balance the influence of poisoned gradients and maintain accuracy, using only true class data for retraining. Sequential removal of multiple points is managed through a simulated queue of redaction requests. In cases where removal impacts task accuracy, a brief additional training phase with new or original data helps in recovery while ensuring continued compliance with data privacy regulations.

Metric

The technique achieves significant time savings, being approximately 10 times faster than the process of removing sensitive data points and retraining the model. This efficiency advantage becomes more pronounced with larger datasets or longer training epochs, where this technique remains unaffected in terms of time required. Moreover, the approach

maintains task accuracy while effectively reducing membership inference confidence to ensure that removed points are consistently misclassified as "Out" or not seen in training.

6.1.3. Fast Yet Effective Machine Unlearning

Definition

The error-maximizing noise technique, as proposed in [9], involves the generation of noise patterns tailored to induce misclassification in the forget set while preserving the model's performance on the retain set. This technique is formulated as an optimization problem aiming to find the $\mathcal{N}$ that maximizes the $L(M, y)$ for the forget classes while minimizing the magnitude of the noise. The optimization process typically involves techniques such as gradient descent or stochastic gradient descent to iteratively update the w_{noise} of the $\mathcal{N}$ until convergence. Then, the noise matrix $\mathcal{N}$ is applied to the forget set during the impair step of the unlearning process. Finally, the repair step in the unlearning process involves fine-tuning the model on a retain set to recover its performance on the remaining classes.

Metric

The proposed method demonstrates superior performance in unlearning specific classes compared to baseline methods, such as FineTune and NegGrad. It effectively reduces the accuracy on the forget set to near zero while maintaining high accuracy on the retained set. The method also shows comparable weight distance to retraining, suggesting effective modification of network weights without overfitting to noise.

6.1.4. Mnemonic Code

Definition

The Learning with Selective Forgetting technique [10] proposes to forget specified classes while preserving others selectively. The core of this technique involves the use of mnemonic codes, which are unique, synthetic signals assigned to each class. These mnemonic codes are generated as random pixel value images for each class and are embedded into training samples to create augmented samples. The embedding process for a sample x_i^k of class c in task k involves generating the augmented sample $\tilde{x}_i^k$ as follows:

$$\tilde{x}_i^k = \lambda x_i^k + (1 - \lambda)\xi_{k,c} \tag{3}$$

where λ is a random variable in $[0, 1]$ and $\xi_{k,c}$ is the mnemonic code for class c.

The model is trained using a total loss function that consists of four terms: classification loss, mnemonic loss, selective forgetting loss, and a regularization term. The classification loss would be softmax cross entropy or additive margin softmax. The mnemonic loss ties each mnemonic code to the corresponding class using the augmented samples. The selective forgetting loss ensures that only classes in the preservation set are remembered. The regularization term L_R prevents catastrophic forgetting using adapted versions of existing regularization methods: Learning without Forgetting, Elastic Weight Consolidation, and Memory-Aware Synapses. After training, only the mnemonic codes for the preservation set classes are retained. The mnemonic codes for the classes in the deletion set are discarded to ensure they are forgotten.

Metric

Across datasets, the technique achieves an average accuracy of approximately 0.90 for preservation sets, which is notably higher compared to [8,9], ranging from 0.80 to 0.85. It demonstrates robustness across varying ratios of classes in deletion sets, showcasing consistent performance across different task complexities and dataset compositions.

Architecture Based

Architecture-based unlearning techniques leverage modifications in the model architecture to facilitate the unlearning process. These techniques can be further categorized into modular unlearning, gradient ascent, teacher–student models, and scrubbing weights, each focusing on restructuring or modifying the model's architecture to facilitate targeted data removal while preserving overall model performance.

6.2. Architecture Based: Modular Unlearning

These methods focus on enhancing model adaptability by facilitating selective data removal without the need for model retraining. Each technique introduces unique strategies tailored to mitigate the impact of removing specific data points from trained models. Through innovative partitioning and isolation strategies, these approaches aim to preserve model accuracy while minimizing computational overhead associated with traditional retraining methods. The subcategory modular unlearning will contain specific notation; please refer to Table 4 for symbols and definitions.

Table 4. Specific notation for SISA approach.

Symbol	Description
S	Number of shards
R	Number of slices per shard
K	Number of unlearning requests
$1/S$	Fraction of the data used for training
D_i	Shard i
$D_{i,j}$	Slice j of shard i
G	Encoding matrix
D_i	Dataset block i
d	Block index

6.2.1. SISA Original

Definition

SISA [11] accommodates unlearning requests by facilitating targeted model updates based on the removal of specific data points. At its core, SISA leverages the division of the dataset into shards, each representing a distinct subset of the data. Within each shard, the data are further partitioned into slices, allowing for incremental training over successive portions of the dataset. The training process occurs independently for each model, ensuring isolation between models and preventing the exchange of information or updates. This isolation preserves the influence of each shard on its corresponding model, enhancing model specificity and reducing interference from unrelated data points. During inference, predictions from models are aggregated, typically employing strategies like majority voting or averaging, to generate a final prediction. The rationale behind the unlearning is that when you need to unlearn specific data points, you only need to retrain the models corresponding to the shards that contain those data points, not the entire model. This reduces the amount of data that need to be processed during retraining, thus saving time.

Metric

SISA training achieves a desired speed-up for a fixed number of unlearning requests, requiring retraining of only 0.003% of the total dataset size. The efficacy of SISA training

exhibits variability contingent upon dataset characteristics and task intricacy. Instances characterized by imbalanced class distributions or substantial noise levels pose challenges for SISA training, potentially leading to diminished model accuracy. Furthermore, this approach requires significant storage capacity.

6.2.2. Adaptive Machine Unlearning

Definition

The SISA algorithm is known for its robustness against non-adaptive deletion sequences. This means SISA relies on the implicit assumption that the points that are deleted are independent of the randomness used to train the models. Ref. [12] proposes an extension of SISA to handle adaptive deletion requests. These are requests that change dynamically based on user observations or changes in the underlying model. The authors of [12] suggest that by obscuring the internal state of the algorithm using techniques from differential privacy [46], such guarantees can be achieved. The paper leverages the principles of differential privacy to design the enhanced version of the SISA algorithm. Specifically, it ensures that the algorithm's behavior remains indistinguishable under various scenarios induced by adaptive deletion requests. Consequently, it furnishes data deletion guarantees that withstand adversaries with knowledge of the internal state of the machine learning algorithm.

Metric

The evaluation focuses on deletion guarantees. These guarantees measure how well the model maintains accuracy, parameter stability, and information security after selective data removal. These guarantees are represented by metrics such as α, which measures accuracy loss post-deletion; β, indicating changes in model parameters; and γ, assessing residual information leakage about deleted data. Additionally, the paper employs differential privacy metrics ϵ and δ to quantify the level of privacy protection against analyses of model outputs and update sequences. Compared to Standard SISA, the proposed technique improves privacy guarantees by 15% on α, 10% on β, and 12% on γ. Compared to the naive approach, it shows improvements of 20% on α, 18% on β, and 15% on γ.

6.2.3. No Matter How You Slice It

Definition

The paper [13] highlights the tendency of SISA to exacerbate performance disparities between majority and minority classes. They investigated the impact of various imbalance ratios (1:10, 1:100, 1:1000) and different methods to mitigate class imbalance (random over-sampling, random under-sampling, cost-sensitive learning, focal loss, label distribution-aware margin) on SISA, monolith baseline and Rus to 1/sqrt(S). The authors suggest that the RUS baseline, which involves down-sampling the dataset to a shard size of 1/sqrt(S), consistently outperforms SISA and the monolith baseline. Its advantage becomes more pronounced as the imbalance ratio increases while maintaining the same average-case retraining speed-up for unlearning requests. Furthermore, they conduct experiments with different numbers of slices (3, 6, 12 slices) and shards (monolith, 5, 10, 20 shards) to further explore this relationship. The results indicate that the number of shards influences model performance, whereas varying the number of slices has a lesser impact. In this paper, it is also mentioned that certain groups of the population (upper-class young people) are more likely to be aware of privacy rights and, hence, more probable to request data deletion. Consequently, the authors recognized the importance of distribution-aware sharding, which involves sorting samples based on their likelihood of being forgotten, to optimize the unlearning process.

Metric

The paper uses error rates as the primary metric to evaluate the technique, particularly focusing on the disparity in performance between majority and minority classes. The evaluation compares the SISA technique to a baseline involving random under-sampling (RUS) to a shard size of $\frac{1}{S}$. The paper reports that the RUS baseline consistently outperforms SISA in terms of minority class error rates, with the performance gap increasing as the imbalance ratio rises. For example, with an imbalance ratio of 1:1000, the RUS baseline shows a lower error rate for minority classes compared to SISA, while preserving the same average-case retraining speed-up for unlearning requests.

The findings suggest that minority class performance suffers when the unlearning likelihood is higher, as these samples are relegated to later slices and receive less attention during training. Conversely, minority class performance improves when associated with a lower-than-average unlearning likelihood. This is because samples with lower unlearning likelihoods are prioritized during training, allowing the model to learn their features more effectively.

6.2.4. Coded Machine Unlearning

Definition

The framework [14] proposes preprocessing the training dataset. This process involves generating a G using the RandMatrix function. Each entry g_{ij} of G represents whether the samples from the i-th shard contribute to the j-th coded shard. Then, each coded shard is sent to a weak learner that trains on this subset of data; finally, the master node aggregates the models. The unlearning algorithm operates on the encoded dataset but uses information about the original unencoded samples to identify the relevant shards and update the model accordingly. The encoding matrix G is used to map between the original and encoded representations of the dataset. This method enables more efficient unlearning while exhibiting a better trade-off in terms of performance in terms of MSE versus unlearning cost. The protocol is designed to handle large-scale datasets efficiently, making it scalable to real-world applications with extensive data volumes.

Metric

This technique compares to retraining from scratch. The technique demonstrates an average improvement of 15% in accuracy. Computational efficiency is enhanced by reducing training time by 30% due to encoded shards and parallelized weak learner training. Moreover, the unlearning cost is reduced significantly, achieving a 75% decrease.

6.2.5. DeepObliviate

Definition

In comparison to previous techniques, DEEPOBLIVIATE [15] divides the dataset only into uniform blocks and trains models independently on each block. Model parameters P_i are saved after training each block D_i to quantify the influence on the model parameters of unlearned data, called "residual memory". DEEPOBLIVIATE first computes the original update vector

$$V_k = P_k - P_{k-1}, \tag{4}$$

representing the change in model parameters from block D_{k-1} to D_k. Subsequently, when retraining without x_u, it computes the retrained update vector

$$V'_k = P'_k - P'_{k-1}, \tag{5}$$

where P'_k and P'_{k-1} are the parameter vectors after retraining on D_k and D_{k-1} without x_u. The quantification of residual memory Δ_k between these vectors, given by

$$\Delta_k = \|V_k - V'_k\|_1 = \|P_k - P_{k-1} - (P'_k - P'_{k-1})\|_1, \tag{6}$$

measures the influence of x_u on model parameters. This approach leverages these differences to decide the point t at which the residual influence of x_u becomes negligible and stop retraining.

Following these calculations, the technique constructs the M' with parameters initialized to P_{d-1}, representing the state of the model parameters up to block D_{d-1} before the block that contains the data to be unlearned D_d is processed. M' is then retrained on the dataset $\{D_{\prime d}, \ldots, D_{d+t}\}$, where $D_{\prime d}$ excludes x_u. To integrate the effects of the remaining blocks $\{D_{d+t+1}, \ldots, DB\}$ into M', the authors employ model stitching, where M' is adjusted as follows: $M' \leftarrow M' \oplus (M \ominus M_{d+t})$. Here, M_{d+t} signifies the model state after training up to block D_{d+t}, where the residual memory of x_d is considered negligible.

Metric

Under the same experimental conditions, DEEPOBLIVIATE achieves superior results compared to SISA, including a 5.8% increase in accuracy, 1.01x faster retraining, and a 32.5x faster prediction speed, all while maintaining equivalent storage requirements across the datasets evaluated.

6.2.6. ARCANE

Definition

Instead of uniformly dividing the dataset D, ARCANE [16] partitions it based on class labels. This means that each subset D_i contains instances belonging exclusively to a single class i. This approach ensures that models trained on each D_i can focus specifically on learning and distinguishing features relevant to that particular class. ARCANE employs information theory principles, such as entropy calculations to identify instances belonging to class i while treating all other instances as anomalies. After individual one-class classifiers make their predictions, the final output is the class with the lowest anomaly score. This score indicates the highest confidence that the sample belongs to that class. ARCANE aligns to SISA to ensure a fair comparison between these two methods in the following way: ARCANE's parameter $m = 20$ (block number) was aligned with R (slice number) in SISA. The number of sub-models in ARCANE was equivalent to shard in SISA.

Metric

ARCANE demonstrated faster retraining times than SISA. ARCANE also maintained competitive accuracy levels and excelled in handling unbalanced training data. In contrast, SISA requires balanced data shards.

6.3. Architecture Based: Gradient Ascent

Gradient ascent techniques in machine unlearning represent a strategic reversal of the traditional gradient descent process used in training machine learning models. Instead of minimizing the loss function, these methods aim to increase it, effectively removing the influence of specific data points or patterns from the model. This section reviews several notable techniques that utilize gradient ascent to achieve unlearning, examining their methodologies and effectiveness. The subcategory gradient ascent will contain specific notation, please refer to Table 5 for symbols and definitions.

Table 5. Specific notation for gradient ascent approach.

Symbol	Description
e, E	Epoch
b, B	Batch
s	Sensitive Data
η	Learning rate

6.3.1. Amnesiac Machine Unlearning

Definition

The paper [17] proposes "Amnesiac Unlearning". Amnesiac unlearning seeks to precisely remove the impact of sensitive data from a neural network by reversing specific parameter updates made during the training process. The methodology involves tracking parameter updates $\Delta\theta_{e,b}$ for each batch in each epoch during training. Batches b_S that contain sensitive data are identified, and a list of these parameter updates $\Delta\theta_{sb}$ is maintained. To perform unlearning, the model parameters are adjusted by removing the influence of these specific updates. Mathematically, let the initial model parameters be θ_{initial}, and the parameters after training for E epochs, each consisting of B batches, are given by

$$\theta_M = \theta_{\text{initial}} + \sum_{e=1}^{E} \sum_{b=1}^{B} \Delta\theta_{e,b} \tag{7}$$

To unlearn, the model parameters are adjusted as follows:

$$\theta_{M'} = \theta_M - \sum_{sb \in SB} \Delta\theta_{sb} \tag{8}$$

The resulting parameters $\theta_{M'}$ exclude the influence of the sensitive data.

Metric

Paper [17] evaluates the proposed amnesiac unlearning technique using three metrics: accuracy, model inversion attacks, and membership inference attacks. This technique is compared against naive retraining. For test accuracy, amnesiac unlearning quickly reduces accuracy on data that are intended to be unlearned, unlike naive retraining, which maintains high accuracy for several epochs. In model inversion attacks, naive retraining fails to prevent information leakage, while amnesiac unlearning significantly obscures sensitive information. In membership inference attacks, naive retraining shows only a gradual reduction in recall, remaining effective for two epochs, whereas amnesiac unlearning reduces recall to near zero immediately.

6.3.2. Unrolling SGD

Definition

Paper [18] introduces a method to reverse the effect of a specific data point x_u on the model by adding back the gradients associated with x_u. The process begins with the model computing predictions for the target data point x_u through a forward pass, generating output logits based on the input data point. Once the forward pass is complete, the gradient of the loss function concerning the model weights W is computed through backpropagation. This gradient, denoted as $\frac{\partial L}{\partial W}$, represents the sensitivity of the model's predictions to changes in the weights.

To perform the unlearning process, the computed gradient adjustment is then added back to the current weights W_t. This adjustment aims to exclude the influence of x_u from the

model's predictions. The learning rate, batch size, and the number of epochs are employed to update the model weights accordingly:

$$w_{t+1} = w_t + \eta \frac{e}{b} \frac{\partial L}{\partial W}\bigg|_{W_0, x_u} \tag{9}$$

This iterative process allows the model to adapt and effectively unlearn the effect of x_u without necessitating complete retraining from scratch.

Metric

This primary study introduces a new metric, called unlearning error. The unlearning error is defined as the Euclidean distance between the model weights after training for t steps and the initial model weights.

$$\text{Unlearning Error} = \|W_t - W_0\|_2 \tag{10}$$

The unlearning error specifically examines the impact of a data point $\mathbf{x}_u$ on the final weights of the model when training begins at initial weights $\mathbf{W}_0$. It is defined to approximate the verification error. Verification error involves comparing the terminal weights of a naively retrained model with the weights of an approximately unlearned model to assess the degree of unlearning. The calculation of verification errors can be resource-intensive due to the need to retrain a model from scratch. The paper compares the cost-effectiveness of their approximate unlearning method with SISA—the cheapest exact unlearning method. They find that their method, which only requires computing a single gradient, is more efficient and less storage-intensive than [11].

6.3.3. BAERASER

Definition

The BAERASER framework [20] introduces a machine unlearning process designed to forget data that trigger backdoor attacks on machine learning models. It begins with trigger pattern recovery, where a max-entropy staircase approximator is utilized to generate and identify potential trigger patterns within the victim model. Once the trigger patterns have been identified, the machine unlearning process is initiated to erase these patterns from the model's memory. This process uses gradient ascent optimization to adjust the model parameters, effectively reversing the influence of the backdoor attack. The optimization is formulated as

$$\theta_{j+1} = \theta_t + \frac{\partial L}{\partial \theta_t} \tag{11}$$

The loss function for machine unlearning incorporates both the cross-entropy loss and a penalty mechanism to prevent over-unlearning. The loss function is defined as

$$\begin{aligned} L = \alpha(L_{CE}(F(\theta_t(x_c), y_c)) - L_{CE}(F(\theta_t(x_u), y_u))) \\ + \beta \sum_{k=1}^{M} W_x \left|\theta_j(x) - \theta_0(x)\right| \end{aligned} \tag{12}$$

Here, L_{CE} denotes the cross-entropy loss function, (x_c, y_c) represents the clean validation data, and (x_u, y_u) represents the trigger pattern data aimed to be forgotten. The parameters α and β are coefficients that balance the degrees of unlearning and penalty, respectively. The weights W_x for each parameter dimension are computed to correlate the penalty with the model's performance on the validation data.

Metric

The BAERASER unlearning technique is evaluated using attack success rate (ASR) and model accuracy (Acc) as primary metrics. ASR measures the percentage of poisoned data misclassified into the attacker's desired target label, while Acc gauges the overall accuracy of the model on clean data. BAERASER's performance is compared to three existing backdoor defense methods: fine-pruning, fine-tuning, and neural attention distillation. Experimental results show that BAERASER outperforms these baselines. BAERASER reduces ASR from nearly 100% to about 10% across various datasets, indicating a marked improvement in backdoor defense effectiveness. Additionally, BAERASER maintains less than a 10% drop in Acc, demonstrating its ability to minimize accuracy loss while significantly lowering ASR.

6.3.4. Forsaken

Definition

At the core of Forsaken [19] lies the mask gradient generator G. Given the current model parameters θ and predictions on the samples marked for forgetting, G generates mask gradients δ that indicate the necessary adjustments to the model parameters to facilitate forgetting. These mask gradients serve as directional cues, guiding the model's updates to selectively remove the specified information associated with the forgotten samples while ensuring minimal disruption to the model's performance on other tasks.

The unlearning process in Forsaken unfolds iteratively over a series of steps aimed at refining the model's behavior. Once the mask gradients are generated, they are used to update the model parameters θ, nudging the model towards a state where the specified information becomes less influential in its predictions. To quantify the discrepancy between the model's predictions for the forgotten samples and a predefined distribution of non-member data, Forsaken employs KL divergence D_{KL} as a measure of dissimilarity. By minimizing the D_{KL} loss, the model's behavior on the forgotten samples gradually aligns with that of non-member data, effectively "forgetting" the specified information. To prevent overfitting during the unlearning process, Forsaken incorporates R_{L1} into the optimization objective. This regularization term penalizes large parameter values, promoting smoother updates to the model parameters and guarding against drastic changes that could compromise the model's performance.

Metric

In addition to standard performance metrics such as accuracy, precision, recall, and F1-score, the paper introduces a metric known as the forgetting rate. It provides a quantitative measure of the rate at which samples transition from being classified as members of the training set to non-members after the unlearning process. A higher forgetting rate indicates a more effective unlearning method, as it signifies a greater reduction in the model's reliance on memorized information. Experimental results show that Forsaken achieves a significantly higher forgetting rate compared to existing techniques (SISA, Full retraining, and SMU), indicating its ability to selectively forget specific information.

6.4. Architecture Based: Teacher–Student

The teacher–student framework is widely used in machine unlearning methods, where a well-trained teacher model guides a student model to shed specific knowledge. Initially, the teacher model, pretrained on the complete dataset, and the student model, initialized either randomly or with the teacher's parameters, are established. Additional models like generators may be used to create synthetic data. Tailored loss functions, such as Kullback–Leibler divergence and cross-entropy loss, are then defined to guide the unlearning process.

The student model undergoes training or fine-tuning with these loss functions to either retain or remove specific knowledge, ensuring alignment with the unlearning objectives. The following subcategory will contain specific notations. Please refer to Table 6 for symbols and definitions.

Table 6. Specific notation for teacher–student approach.

Symbol	Description
KL	Kullback–Leibler (KL) divergence
θ	Random weights
JS	Jensen–Shannon divergence
$\hat{y}_i$	Predicted probability distribution for the i-th data point

6.4.1. Bad Teaching

Definition

The proposed unlearning method [21] utilizes a teacher–student framework with two types of teachers: competent and incompetent. The competent teacher $T_s(x;\theta)$ has learned from the complete dataset D, while the incompetent teacher $T_d(x;\phi)$ is a smaller model initialized with random weights. The student model $S(x;\theta)$ is initialized with the same parameters as the competent teacher. The student model is trained to mimic the predictions of the incompetent teacher for the data points that need to be forgotten D_f via minimizing the Kullback–Leibler (KL) divergence. This helps the student "forget" these data points. Simultaneously, the student is trained to retain accurate knowledge by mimicking the predictions of the competent teacher for the data points that should be retained D_r. Mathematically, this objective is expressed as

$$\begin{aligned} L(x, lu) = (1 - lu) \cdot \mathrm{KL}(T_s(x;\theta) \,||\, S(x;\theta)) \\ + lu \cdot \mathrm{KL}(T_d(x;\phi) \,||\, S(x;\theta)) \end{aligned} \tag{13}$$

where lu is the unlearning label.

Metric

Compared to amnesiac unlearning [17], bad teaching achieves lower activation distance and maintains higher accuracy on forget sets across various datasets. Amnesiac unlearning damages forget set performance significantly, indicating the Streisand effect, while bad teaching does not. In addition to traditional metrics, this technique introduces a metric called the Zero Retrain Forgetting Metric (ZRF). ZRF measures the randomness in the model's prediction by comparing them with the incompetent teacher's predictions. The ZRF score improves after unlearning with the technique, indicating effective forgetting without needing a reference retrained model. For example, the ZRF score of the model increases from 0.87 to 0.99 after unlearning. Furthermore, the JS-Divergence between the predictions of the unlearned model and the retrained model is low, indicating that the output distribution of the unlearned model is very close to the model retrained from scratch. Additionally, the probability of a successful membership inference attack on the forgotten set decreases significantly after unlearning. For instance, in the case of forgetting rocket images, the attack probability drops from 0.982 to 0.002, indicating improved privacy.

6.4.2. Gated Knowledge Transfer

Definition

Gated knowledge transfer [22] is a zero-shot type of unlearning. Its process begins with the initialization of three components: the teacher model M_T, the student model

$M_S(x;\theta)$, and the generator $G(z;\phi)$. The teacher model is the pretrained model from which knowledge is to be transferred. The student model $M_S(x;\theta)$, with the same architecture as the teacher, starts with random initialization. The generator $G(z;\phi)$ also begins with random parameters and is responsible for creating pseudo samples from noise vectors.

Once initialized, the generator produces pseudo samples by transforming noise vectors $z \sim N(0, I)$. These pseudo samples serve as synthetic data points focused on the transferal of knowledge from the teacher model to the student model. To achieve this, the generator is updated to maximize the KL-divergence between the teacher's and student's output distributions for the filtered pseudo samples. On the pseudo samples, a band-pass filter is applied with the intention to not convey information about the forgotten classes. The filter works by checking the teacher's predicted probabilities and allowing a pseudo sample to pass only if the predicted probability for each forget class is less than a threshold ϵ. The threshold parameter ϵ must be low enough to filter out information about the forget classes effectively but not so low that no samples pass through the filter. This is stated to prevent any significant information about the forget classes from being transferred to the student model.

The student model is updated to minimize a combined loss function. This loss function comprises the KL-divergence between the teacher and student models' outputs and an attention loss. The attention difference serves as a mechanism to encourage the student model to focus on the same features as the teacher model, thus facilitating effective knowledge transfer. The generator and the student model are updated iteratively. The generator aims to create pseudo samples that maximize the divergence between the teacher and student, while the student seeks to minimize this divergence and learn effectively from the teacher's knowledge, except the forgotten classes. This iterative process continues until the models converge. The student model retains knowledge of the retained classes while forgetting the specified classes.

Metric

The gated knowledge transfer (GKT) technique proposed in this paper was evaluated against several established methods, including Fisher forgetting (FF) [27], amnesiac unlearning (AU) [17], and the retrain baseline (RB). The GKT method achieved a significantly lower Anamnesis Index (AIN) value; this metric is calculated based on the speed of relearning (how quickly the model can regain knowledge). For example, the GKT method's AIN was 0.1 compared to 0.3 for FF and 0.25 for AU. In terms of accuracy on the forget set, the GKT method consistently achieved near 0% accuracy, indicating that the target information was forgotten. On the retained set, the GKT method maintained high accuracy, achieving 82% and showing competitive performance while ensuring unlearning.

6.4.3. Efficient Two-Stage Model

Definition

The proposed technique [23] begins by computing the model output for each data point within a specified subset. It then identifies pairs of classes with the largest divergence or discrepancy in their output probabilities. During the training phase, data points in the designated subset are intentionally mislabeled with classes that are most different from their true labels. Training persists until the model's accuracy on subset P descends below random prediction thresholds. Following the neutralization phase, the subsequent stage involves knowledge distillation (KD), where the teacher–student relationship is established. Here, the knowledge from the original teacher model is distilled into the student model M'. KD facilitates the emulation of information from the teacher model by softening label

probabilities within M'. The soft label knowledge distillation loss is represented by the following equation:

$$\mathcal{L}_{KD} = \sum_{i=1}^{N} \delta(\hat{y}_i)\delta\left(\log \frac{\hat{y}_i}{y_i}\right) \tag{14}$$

where $\hat{y}_i$ is from the teacher model and $\delta(\hat{y}_i)$ is the softened predicted probability distribution. This encourages M' to generate output probabilities akin to those of the teacher model, thus fostering knowledge transfer. Concurrently, cross-entropy loss provides an additional training signal to augment the learning process. The combined loss function used for training M'_D integrates both KD and cross-entropy loss, expressed as

$$\mathcal{L}_{TOTAL} = \alpha\mathcal{L}_{CE} + \beta\mathcal{L}_{KD} \tag{15}$$

Metric

The proposed method is evaluated using accuracy as the primary performance metric. It compares the performance of three models: the original model, the retrained model using the proposed technique, and a scratch model. The student model achieves 65.25% accuracy compared to the model retrained from scratch, which reaches 64.43% accuracy on the remaining data, showcasing an improvement of 0.82%.

6.4.4. Towards Unbounded Machine Unlearning

Definition

The paper [24] proposes the SCRUB method, where the original model, referred to as the teacher model, is trained on the full dataset. The student model starts with the weights of the teacher model. This methodology also uses the KL-divergence between the output distributions of the teacher and student models. The optimization objective for the student model is formulated to minimize the following function:

$$\begin{aligned} \min_{w_u} \alpha \frac{1}{N_r} \sum_{x_r \in D_r} & d(x_r; w_u) \\ + \gamma \frac{1}{N_r} \sum_{(x_r, y_r) \in D_r} & \mathcal{L}_{CE}(f(x_r; w_u), y_r) \\ - \frac{1}{N_u} \sum_{x_u \in D_f} & d(x_u; w_u) \end{aligned} \tag{16}$$

where α and γ are hyperparameters, d is a distance function, N_r is the number of examples in the retain set, and N_f is the number of examples in the forget set. The student model undergoes an alternating optimization process. Training alternates between updating the student model on the forget set (max-step) and the retain set (min-step). Additional min-steps are performed at the end of the sequence to ensure the retain set performance is restored. Training stops when the forget set error has increased sufficiently without harming the retain set error.

SCRUB+R extends SCRUB by incorporating a "rewinding" procedure to address vulnerabilities to membership inference attacks (MIAs). A reference point for the forget error is established by constructing a validation set that has the same distribution as the forget set. SCRUB is then trained while storing model checkpoints at each epoch. At the end of training, the validation set error is measured to serve as the reference point for the desired forget set error. The rewinding procedure involves rewinding to the checkpoint where the forget error is closest to the validation set error, ensuring that the forget set error is "just high enough" to prevent MIAs.

Metric

In this study, the evaluation of the SCRUB and SCRUB+ unlearning methods is conducted using three distinct sets of forget-quality metrics tailored to specific applications: Removing Biases (RB), Resolving Confusion (RC), and User Privacy (UP). The methods are compared against state-of-the-art approaches, including Retrain, [21,27,28]. Across the RB scenarios, SCRUB demonstrates robust performance, achieving an average forget error of 78.4%, outperforming the next best method ([21]) by 15.6%. In RC scenarios, SCRUB exhibits an average reduction in interclass confusion error of 63.2%, surpassing its closest competitor (retain baseline) by 12.8%. Notably, in UP scenarios, SCRUB+ showcases improvements, with a 45.9% decrease in membership inference attacks compared to the strongest baseline ([21]).

6.4.5. Deep Regression Unlearning

Definition

The Blindspot Unlearning technique [26] is a method devised for the selective removal of information from deep regression models. It operates through a collaborative optimization process involving two distinct models: the Original Fully Trained Model and the Blindspot Model. The Blindspot Model is initialized randomly and exposed partially to samples solely from the retain set. It functions as a reference for output distribution and activation closeness comparisons with the Original Fully Trained Model. The optimization process integrates three distinct loss functions: loss computation for the retain set samples in the Original Fully Trained Model (L_r), loss evaluation by contrasting output similarities between the Original Fully Trained Model and the Blindspot Model (L_f), and assessment of layerwise activation closeness between both models (L_{attn}). Mathematically, the final loss equation is expressed as

$$L = (1 - l_{if})L_r + l_{if}(L_f + L_{attn}) \tag{17}$$

where $l_{if} = \begin{cases} 1 & \text{for samples in the forget set} \\ 0 & \text{otherwise} \end{cases}$.

Minimize the combined loss function L through gradient-based optimization techniques. This optimization process updates the parameters ϕ of the Original Fully Trained Model to selectively remove information related to the forget set while retaining the information pertinent to the retained set.

Metric

In comparison to baseline methods such as fine-tuning and gradient ascent baseline methods, the Blindspot Unlearning technique outperformed both. Finetune on the retain dataset led to catastrophic forgetting on the forget set, while NegGrad resulted in the Streisand effect. The Blindspot Unlearning technique provided error rates on the forgotten set that were similar to those of the retrained model. This technique presents a lower attack probability, indicating better privacy preservation. Furthermore, it demonstrates a Wasserstein distance metric that aligns more closely with the retrained model. Moreover, the Anamnesis Index values were closest to 1 for the Blindspot Unlearning technique across different datasets and domains, indicating superior unlearning performance.

6.5. Architecture Based: Scrubbing Weights

The scrubbing weights approach comprises a category of machine unlearning techniques dedicated to modifying weights to diminish the influence of selected data points or datasets. These methods leverage rigorous mathematical frameworks such as Hes-

sians, Fisher information matrices (FIM), and their approximations to achieve targeted data removal. By applying strategic transformations and introducing controlled noise into the weight space, these techniques facilitate selective forgetting while preserving essential model knowledge. This approach aims to enhance model robustness, privacy, and adaptability in dynamic learning contexts. The following subcategory will contain specific notations. Please refer to Table 7 for symbols and definitions.

Table 7. Specific notations for scrubbing weights approach.

Symbol	Description
$S(\theta)$	Scrubbed model parameters
λ	Hyperparameter controlling forgetting
σ	Error in approximating the SGD behavior
h	Transformation function
F	Fisher information matrix (FIM)
n	Noise
B^{-1}	Inverse of the Hessian matrix
w_u	Linear user weights
L_{MSE}	Mean square error loss

6.5.1. Eternal Sunshine

Definition

Ref. [27] proposes a selective forgetting procedure tailored for Deep Neural Networks trained with stochastic gradient descent. The core of the forgetting mechanism involves a shift in weight space and the addition of noise to the weights. Furthermore, the paper provides an upper bound on the amount of remaining information in the weights of the network after applying the forgetting procedure. This suggests that the proposed forgetting mechanism has a quantifiable effect on reducing the information stored in the model weights, with an upper limit on the residual information. The optimal scrubbing procedure is represented in the form

$$S(\theta) = h((\theta) + n \tag{18}$$

where

$$h((\theta) = (\theta - (B^{-1}\nabla L_{Dr}((\theta)) \tag{19}$$

where S(θ) are the scrubbed model parameters, h(θ) represents the transformation applied to θ to forget D_u, and n is a noise term following a Gaussian distribution with mean 0 and covariance matrix Σ. This has two variations: Fisher forgetting and variational forgetting. In the first case, the Hessian is approximate with the diagonal of the Fisher information matrix or a better Kronecker-factorized approximation. So, the equations is as follows:

$$S(\theta) = \theta + (\lambda\sigma^2 h)^{\frac{1}{4}} F^{-\frac{1}{4}} \tag{20}$$

In the second case, instead of computing the FIM, the noise is optimized in the Forgetting Lagrangian. The author minimizes the proxy:

$$L(\Sigma) = \mathbb{E}_{n\sim\mathcal{N}(0,\Sigma)} L_{Dr}(\theta + n) - \lambda \log|\Sigma| \tag{21}$$

Further, the optima Σ is seen as the FIM computed.

Metric

The paper evaluates its technique using several metrics: error on the forgotten cohort D_u, error on the remaining data D_r, relearn time measured in epochs, and an information upper-bound on retained information. It compares its approach against fine-tuning, negative gradient, random labels, and hiding methods. Results indicate reductions in error on D_u and D_r, slower relearn times, and lower information bounds compared to alternative methods.

6.5.2. Forgetting Outside the Box

Definition

Paper [28] extends the selective forgetting framework to consider activations (output of intermediate layers) rather than just weights as in [27]. It introduces a technique called NTK-based scrubbing, which leverages insights from the Neural Tangent Kernel theory to improve selective forgetting. The process begins by linearizing the final activations around pretrained weights. This involves computing the linear approximation of the final activations using gradients. Using the linearized activations, the optimal forgetting function is computed. This function represents the transition from the weights trained on the complete dataset to the weights that would have been obtained by training on the retained dataset alone. Mathematically, the optimal forgetting function can be expressed as follows:

$$h_{\text{NTK}}(\theta) = \theta + P\nabla f_0(D_f)^T MV \tag{22}$$

where

- P is a projection matrix that projects the gradients of the samples to be forgotten onto the orthogonal space to the space spanned by the gradients of all samples to be retained;
- $\nabla f_0(D_f)^T MV$ is the matrix whose columns are the gradients of the samples to forget, computed at θ_0.

The final scrubbed weights ($S_{\text{NTK}}(w)$) are obtained by combining the optimal forgetting function ($h_{\text{NTK}}(w)$) with the noise (n). $S_{\text{NTK}}(w)$ represents the updated weights of the network after the selective forgetting process. This process discards outdated or irrelevant information while preserving important knowledge. Noise (n) is added to the optimal forgetting function to increase robustness and prevent the network from overfitting the specific features of the data.

Metric

They use the same readout functions of [27] and add a black-box membership inference attack. In error readout analysis, NTK demonstrates superior performance by minimizing error rates on both retain and test sets compared to Fisher forgetting, which requires excessive noise addition due to large weight space distances. Additionally, NTK surpasses baselines in relearn time, indicating its efficacy in reducing remaining information about the forgotten cohort. Robustness against black-box membership inference attacks further highlights NTK's superiority, achieving optimal accuracy while Fisher forgetting risks undesired information leakage.

6.5.3. Mixed Privacy

Definition

The paper [29], instead of linearly approximating the training activation, as stated before, proposes to train a linearized network directly for forgetting. The goal is to transform the original deep neural network into a mixed-linear model, which is a combination of

non-linear core weights w_c and linear user weights w_u. This mixed-linear model, which can be seen as a first-order Taylor approximation of the effect of fine-tuning the original deep network, is formulated as follows:

$$f_{ML}(w_c^*, w_u)(x) = f_{w_c^*}(x) + \nabla_w f_{w_c^*}(x) \cdot w_u \tag{23}$$

Here,

- $f_{w_c^*}(x)$ represents the output of the original deep network with the core weights w_c^*;
- $\nabla_w f_{w_c^*}(x)$ represents the gradient of the output with respect to the core weights w_c^*, evaluated at x.

The training of the mixed-linear model involves solving two separate minimization problems:

1. Training Core Weights w_c^*:

$$w_c^* = \text{argmin}_{w_c} L_{CE}(f_{w_c}) \tag{24}$$

2. Training User Weights w_u:

$$w_u^* = \text{argmin}_{w_u} L_{MSE}(f_{ML}(w_c^*, w_u)) \tag{25}$$

By transforming the original DNN into a mixed-linear model, the authors aim to facilitate the forgetting process. The optimal forgetting step to delete D_f is given by

$$w_u \mapsto w_u - H_{Dr}^{-1}(w_c)\nabla_{w_u} g_{Dr}(w_u), \tag{26}$$

The forgetting update is formulated as the optimal adjustment of w_u, achieved by computing the inverse of the Hessian matrix of the loss function for the core weights w_c evaluated on the remaining data D_r and applying the gradient of the loss function concerning w_u. Since computing the full Hessian matrix is impractical, an auxiliary loss function $\hat{L}_{D_r}(v)$ is introduced. Finally, to enhance stability and ensure robust forgetting, random noise is added to the weights.

Metric

The readout functions include error rates on subsets of data, relearn time, activation distance, and membership attack success. Activation distance quantifies the difference in final activations between scrubbed and retrained models, providing insight into the residual information about the forgotten data. The paper compares the proposed method, ML-forgetting with Fisher forgetting. ML-forgetting outperforms other methods, particularly in reducing relearn time and activation distance.

6.5.4. Certified Removal

Definition

The paper [30] introduces certified removal (CR), tailored initially for convex models but adaptable to non-convex models. For a specific data point within D, C modifies the model output $M(D)$ such that the resultant model $C(M(D), D, x)$ closely approximates the model trained on $D \setminus \{x\}$. This closeness is quantified by a probabilistic condition ensuring that the distributions of outputs under C and under retraining without x are indistinguishable within a specified tolerance ϵ. Mathematically, the CR mechanism C aims to satisfy

$$e^{-\epsilon} \leq \frac{P(C(M(D), D, x) \in T)}{P(M(D \setminus \{x\}) \in T)} \leq e^{\epsilon} \tag{27}$$

where T denotes the set of possible model outputs and $\epsilon > 0$ is a parameter controlling the level of removal certainty. A key component of this mechanism involves a Newton step, leveraging the Hessian of the loss function at the current model parameters θ^*. The Newton update

$$\theta^- = \theta^* + H_{\theta^*}^{-1}\Delta \tag{28}$$

where Δ represents the gradient influence of the removed data point on the model parameters θ^*. The Hessian H_{θ^*} captures the curvature of the loss function around θ^*, providing a quadratic approximation that guides the adjustment of θ^* to θ^-. For deep neural networks and non-convex models, the adaptation involves applying similar principles to the linear decision-making layer.

Metric

This paper evaluates its certified removal technique primarily through metrics of accuracy and computational efficiency. Experiments on sentiment analysis and digit classification tasks using deep neural network feature extractors demonstrate substantial accuracy gains and efficiency improvements compared to fully private models or retraining approaches.

6.5.5. The Projective Residual Update

Definition

The Projective Residual Update (PRU), as introduced in the paper [31], aims to effectively remove specific data points from trained machine learning models. Initially designed for linear models such as logistic and linear regression, PRU's methodology extends to nonlinear models by treating them as comprising a fixed feature mapping followed by a linear or logistic regression layer. This adaptation simplifies the update process by focusing on the linear components of the model's structure, particularly the final layers in deep neural networks.

PRU utilizes synthetic predictions to estimate how the model would predict the outputs for data points earmarked for removal using the current model parameters. These synthetic predictions are pivotal because they act as substitutes for the actual outputs that the model would produce if the identified data points were removed. In the context of linear regression models, for example, these predictions are straightforwardly computed as the dot product of the model's current weights with the feature vector of each data point x_i.

The primary objective of PRU is to adjust the model's current weights so that its predictions for these synthetic outputs closely align with the actual outputs of the removed data points. To achieve this alignment, PRU employs optimization techniques such as gradient descent. Through iterative updates, the model's weights are adjusted based on the disparity between the synthetic predictions and the real outputs of the data points scheduled for removal.

Metric

PRU typically maintains low L2 distances, indicating minimal deviation from exact retraining, especially notable in scenarios with large deletion groups. PRU often shows superior performance in the backdoor injection attack metric compared to [30]. For instance, while [30] might achieve metrics averaging around 0.2, PRU could achieve significantly lower values like 0.05. A lower value in the backdoor injection attack metric indicates that PRU is effective in removing or mitigating the influence of injected features that could compromise privacy.

6.5.6. Performance Unchanged Model Augmentation (PUMA)

Definition

PUMA [32] updates the model parameters θ to θ_{mod}, ensuring minimal disruption to the model's predictive capabilities post-data removal. PUMA's approach uses optimization principles, particularly leveraging the Hessian Vector Product for efficient parameter adjustments. The Hessian Vector Product approximates the impact of changes in model parameters on the loss function gradients, crucial for optimizing θ in response to removed data points.

The technique involves two primary steps: First, PUMA formulates an optimization problem to derive the modified parameters from the original parameters and incorporating adjustments that mitigate the removal of D_f's influence; this step ensures that the model's overall performance criteria are preserved or improved. Second, PUMA optimizes the perturbation factors assigned to the remaining data points $Dr \setminus D_f$; these factors are optimized to minimize the performance degradation caused by removing D_f, balancing between sparsity and small changes using regularization techniques.

Metric

PUMA consistently outperforms traditional methods like the Retrain Model and [11] in several key metrics evaluated in the paper. Specifically, it shows up to a 10% improvement over the original model's performance when assessing the ability to preserve model performance after gradually removing data points. Additionally, in terms of effectiveness in data removal, PUMA reduces the success rate of membership attacks by 20–30% compared to other techniques such as Amnesiac Machine Learning. Furthermore, PUMA demonstrates superior efficiency by executing operations 40–50% faster than competing approaches in scenarios involving random data removal.

6.5.7. Unlearn Features and labels

Definition

Unlearning in [33] involves updating the model parameters when the dataset changes from the original dataset to a modified dataset. This update is achieved using influence functions, a concept from robust statistics that measure the impact of individual data points on the model's parameters. The technique calculates precise updates by using first-order and second-order derivatives to reflect the removal or correction of specific data points or features.

One significant aspect of this approach is its ability to handle feature revocation, which involves removing entire features from the model. The process starts by identifying data points where these features are non-zero, then constructing a modified version of the dataset where these features are set to zero. The model parameters are then adjusted to account for these changes. Despite the reduction in input dimensionality, the method ensures that the model's performance and integrity are maintained through appropriate adjustments derived from the model's linear transformations.

A key consideration in the practical implementation of this technique is its scalability to large and complex models, such as deep neural networks. Direct computation of the Hessian matrix for exact updates is computationally prohibitive in such cases. Therefore, the paper proposes an approximation method for the inverse Hessian matrix. This approach enables efficient second-order updates that balance computational feasibility with maintaining the integrity of the model adjustments during unlearning.

Metric

Compared to traditional baselines like retraining and [11], the proposed technique achieves a 28% improvement in speed while maintaining high fidelity in correcting unintended memorization and label poisoning. The technique corrects poisoned labels, achieving 85% accuracy restoration with 2500 poisoned labels.

6.6. Federated Unlearning

6.6.1. FedEraser

Definition

FedEraser [34] introduces a federated unlearning methodology aimed at reducing the influence of specific client data on a global model within federated learning setups. The primary objective is to adjust the parameters of the global model $\mathbf{w}_{\text{global}}$ to mitigate the impact of individual client contributions without directly accessing or compromising client data privacy. This adjustment process involves iteratively modifying $\mathbf{w}_{\text{global}}$ by subtracting a scaled version of the client's model parameters $\mathbf{w}_c$, denoted as $\gamma \cdot \mathbf{w}_c$, where γ controls the magnitude of adjustment.

To operationalize this unlearning process, FedEraser incorporates a client calibration ratio r, defined as $r = \frac{E_{\text{cali}}}{E_{\text{loc}}}$, where E_{cali} represents the loss of the global model after unlearning and E_{loc} signifies the loss of the client's local model. This ratio guides the extent to which the client's influence is adjusted in the global model, ensuring a balanced approach to privacy preservation and model performance.

Another critical parameter in FedEraser is the retaining interval Δt, which determines the frequency of updates to $\mathbf{w}_{\text{global}}$ during the unlearning process. By carefully selecting Δt, FedEraser maintains stability in the global model while iteratively reducing the influence of client-specific data contributions.

Metric

To compare FedEraser, the paper uses two baselines: FedRetrain, which involves retraining the global model from scratch without the target client's data, and FedAccum, which accumulates updates from multiple clients without specific unlearning. For the Adult dataset, the F1-score for MIAs on the original model is 0.714. After unlearning with FedEraser, the F1-score drops to 0.563, compared to 0.571 for FedRetrain. The impact of the calibration ratio (r) is also assessed. For the Adult dataset, with $r = 0.1$, FedEraser achieves a prediction accuracy of 85.8% on target data in 10.1 s. With $r = 1.0$, accuracy decreases slightly by 0.5%, but time increases to 100.1 s.

6.6.2. FU with Knowledge Distillation

Definition

To eliminate the contribution of a specific client N from the final global model M_F, the paper [35] proposes erasing all historical updates ΔM_i^t from this client for rounds $t \in [1, F-1]$. Given N clients participating in each round t, the global model update ΔM_t can be expressed as

$$\Delta M_t = \frac{1}{N} \sum_{i=1}^{N} \Delta M_i^t \tag{29}$$

To remove the contribution ΔM_N^t of the target client N, the updated model ΔM_0^t is recalculated as

$$\Delta M_0^t = \Delta M_t - \frac{1}{N} \Delta M_N^t \tag{30}$$

Summing up these updates across rounds gives the unlearning version of the final global model M_0^F:

$$M_0^F = M_1 + \frac{N}{N-1} \sum_{t=1}^{F-1} \Delta M_0^t + \sum_{t=1}^{F-1} \epsilon_t \tag{31}$$

where ϵ_t represents the necessary corrections (skew) due to the incremental learning property of FL. This process mitigates skew accumulation caused by earlier model updates.

Knowledge distillation is employed to refine the unlearning model using the original global model M_F as a teacher and the skewed unlearning model as a student. Soft class prediction probabilities q_i are generated using a softmax function over logits z_i:

$$q_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)} \tag{32}$$

where T is a temperature parameter that controls the smoothness of the probability distribution. Higher values of T produce softer distributions, enhancing model generalization. These soft probabilities are utilized to label unlabeled data, effectively transferring knowledge from the original model. During distillation training, if labeled data are available, a weighted average approach is adopted using both hard labels (ground truth) and soft labels produced by the global model at high temperature T. This approach balances the objectives, giving higher weight to soft labels to improve robustness and generalizability. After distillation training, the temperature T is set to 1, refining the unlearning model M_0^F to produce discrete class probabilities suitable for testing scenarios.

Metric

The paper evaluates the proposed unlearning technique using standard metrics to assess its effectiveness in removing the target client's influence from the global model. Comparisons are made against a baseline method of retraining from scratch. Results indicate a reduction in the attack success rate to zero post-unlearning. Additionally, through knowledge distillation, the technique achieves model recovery with test accuracy closely matching that of retraining from scratch.

6.6.3. Pruning

Definition

The federated unlearning technique begins with a pretrained global model that has been developed through a federated learning (FL) framework. In this initial phase, multiple clients participate by training local models on their private datasets and then sharing updates with a central server. This server aggregates these updates to refine the global model, which learns from a diverse and decentralized dataset. Once the global model has been established, the next step involves identifying a specific category, or class, that needs to be removed from the model; this is referred to as the target category.

The core of the unlearning process in [36] is channel pruning. The channels in the neural network that are most associated with the target category are identified and removed. This process involves calculating the relevance of each channel to the target category, which can be achieved using metrics like TF-IDF scores. For instance, the relevance of a channel c_i is measured to determine how significant its contribution is to the target category. Channels with high relevance scores are pruned, meaning their weights W_i are set to zero or otherwise removed from the model. This step ensures that the model's representation of the target category is minimized.

After pruning the channels, the global model undergoes a fine-tuning phase to recover and maintain its performance on the remaining categories. In the federated learning setting, this involves clients training their local models based on the pruned global model. Each

client performs local training iterations and sends model updates back to the central server. The server then aggregates these updates, which are averaged or otherwise combined, to update the global model. This fine-tuning process helps restore the model's accuracy on the non-target categories while keeping the pruned channels effectively removed. The model parameters θ are adjusted using gradient descent methods, where the updates are computed based on the loss function and applied to the model.

Metric

When compared to Fisher unlearning, the federated unlearning technique demonstrated superior performance in terms of accuracy and robustness. Fisher unlearning struggled particularly with high levels of data bias, as it relies on global access to training data to inject optimal noise for unlearning. This limitation led to reduced accuracy due to indiscriminate noise application. In contrast, the proposed technique maintained accuracy on non-target categories (R-Set) and achieved a 0% accuracy on the target category (U-Set), ensuring effective removal of the target class information without significant performance degradation.

6.6.4. Efficient Realization

Definition

The technique detailed in the paper "The Right to be Forgotten in Federated Learning: An Efficient Realization with Rapid Retraining" [37] introduces a sophisticated approach to federated unlearning. Initially, the unlearning process begins with a federated data deletion operation. This results in locally deleted datasets, which contain subsets of the original data with certain samples removed. The process continues with the application of rapid retraining techniques to update the global model in response to the changes in the local datasets. Unlike traditional retraining methods that require exhaustive updates to all model parameters, the proposed technique employs a selective parameter update strategy based on the Fisher information matrix (FIM). By utilizing the FIM, the unlearning process can efficiently compute second-order derivatives necessary for parameter updates while minimizing computational overhead. Furthermore, the technique incorporates momentum techniques to enhance the stability and convergence speed of the unlearning process. The technique has limitations regarding the accuracy of the FIM approximation and the potential for divergence in unstable FL environments. While momentum techniques help mitigate these issues, further research may be needed to address challenges related to approximation errors and model convergence.

Metric

In terms of evaluation metrics, the paper compares the proposed technique to baseline methods such as retraining from scratch. Key metrics include the speed-up factor, which measures the efficiency of the unlearning process, and the Symmetric Absolute Percentage Error (SAPE), which quantifies the difference in model performance between the proposed technique and baseline methods.

6.6.5. FedRecover

Definition

FedRecover [38] is a method designed to recover a federated learning (FL) global model after it has been subjected to poisoning attacks. The first step in FedRecover is the storage of historical data. During each global round, the server stores the model updates submitted by each client. The second step is the detection of malicious clients. At some point, malicious clients are detected based on their submitted updates. Detection mechanisms are not part of FedRecover, but the method assumes that these clients can be identified and removed.

The third step involves the estimation of true model updates. After detecting and removing the malicious clients, the server needs to estimate the true model updates that would have been contributed by non-malicious clients. This estimation process is based on the stored historical updates. The final step is the reaggregation of estimated updates. Using the estimated true model updates, the server performs a reaggregation process similar to the standard federated averaging (FedAvg) but excluding the contributions from detected malicious clients. This reaggregation involves averaging the estimated updates to form a new global model, effectively recovering the model from the poisoning attacks.

Metric

The performance of FedRecover was evaluated using training error rate, which measures the accuracy of the global model, and attack success rate (ASR), which assesses the effectiveness of the attack in altering the model's predictions. FedRecover was compared to the train-from-scratch method and fine-tuning using clean datasets. Results showed that FedRecover achieves training error rate and attack success rate nearly identical to train-from-scratch, even when False Negative Rate is up to 0.5. Specifically, the training error rate curves for FedRecover almost overlap with those for train-from-scratch, except when the False Negative Rate is large (e.g., False Negative Rate ≥ 0.4) for Federated Averaging. Fine-tuning required a large number of clean examples, around 1000 examples, to achieve an training error rate and attack success rate comparable to FedRecover.

6.6.6. SFU

Definition

Federated unlearning involves removing the influence of a specific client, say C_I, from the global model. A straightforward approach might involve maximizing the empirical loss $L_{C_I}(w)$ of the target client C_I, which can be interpreted as reversing the learning process. However, this method would adversely affect the contributions from other clients and degrade the overall model performance. The subspace-based federated unlearning (SFU) technique in [39] addresses this challenge by modifying the gradient updates in a controlled manner. The core idea is to project the gradient ascent updates, which aim to increase $L_{C_I}(w)$, onto a subspace that is orthogonal to the input subspace of the other clients. This ensures that the increase in $L_{C_I}(w)$ only affects the components of the model that are associated with the target client's data, thereby preserving the contributions from other clients.

Mathematically, SFU begins by computing the gradient $\nabla L_{C_I}(w)$ of the target client's empirical loss. This gradient is then restricted to a subspace orthogonal to the input subspace of the remaining clients C_j ($j \neq I$). The updated gradient g_{SFU} can be expressed as

$$g_{SFU} = P_{\perp} \nabla L_{C_I}(w) \tag{33}$$

where $P_{\perp}$ is the projection operator onto the orthogonal subspace of the input space of the remaining clients. The global model update is then performed using this restricted gradient g_{SFU}, effectively removing the contribution of C_I without disrupting the global model's alignment with the other clients' data. The unlearning process in SFU involves iteratively applying this restricted gradient ascent to the global model until the contribution of C_I is sufficiently reduced.

Metric

The performance of the SFU technique was evaluated using two primary metrics: the attack success rate and the model accuracy. SFU reduced the attack success rate to nearly 0%, whereas FU with Knowledge Distillation (UL) and gradient ascent (GA) techniques

achieved rates of 15% and 10%, respectively. This indicates that SFU is highly effective in eliminating the target client's contribution without leaving residual effects.

Regarding model accuracy, SFU maintained high performance after unlearning, with minimal degradation. The model's accuracy with SFU was 98.5%, compared to 42.76% for UL and 92.33% for GA. SFU demonstrated superior efficiency in recovering model accuracy, requiring only one round of training to achieve high accuracy, while UL and GA needed five or more rounds.

6.6.7. KNOT

Definition

The KNOT technique in [40] introduces a solution to the federated unlearning problem by leveraging clustered aggregation combined with asynchronous federated learning. The key innovation in KNOT is the partitioning of clients into clusters, where the global model aggregation is restricted within each cluster. This strategy limits the scope of retraining required when a client requests data erasure, thereby reducing the overall computational cost. Clients $\{C_k\}_{k=1}^{K}$ are partitioned into N clusters $\{L_n\}_{n=1}^{N}$. The server aggregates the local updates $\Delta\omega_k$ from clients within each cluster independently. The global model is thus represented as a combination of cluster-specific models ω_n, where $\omega_n = \frac{1}{|L_n|}\sum_{k\in L_n}\Delta\omega_k$. When a client C_k within a cluster L_n requests data erasure, only the clients in L_n need to retrain, leaving other clusters unaffected. This localized retraining minimizes the ripple effect of unlearning across the entire network. KNOT also employs asynchronous updates to further enhance performance. In asynchronous FL, the server does not wait for all clients to complete their updates; instead, it proceeds with aggregating the updates from faster clients as soon as they arrive. This asynchronous aggregation allows for faster convergence and mitigates the delay caused by slower clients.

A crucial component of KNOT is the optimized assignment of clients to clusters, which is formulated as an optimization problem. The objective is to minimize the wall-clock time required for unlearning by ensuring that clients are assigned to clusters in a way that balances training times and cluster sizes. The problem is framed as a lexicographical minimization problem, where the goal is to minimize the largest value in the match rating vector f given by $\text{lexmin}\, f = (d_{11}x_{11}, \ldots, d_{kn}x_{kn}, \ldots, d_{KN}x_{KN})$, subject to constraints that ensure each client is assigned to a cluster appropriately. This lexicographical minimization problem is then transformed into a linear programming (LP) problem by leveraging the properties of separable convex functions and totally unimodular matrices. The LP problem can be efficiently solved using standard solvers, yielding optimal client–cluster assignments with minimal computational overhead.

Metric

KNOT achieves an impressive 85% reduction in wall-clock training time compared to FedEraser, a state-of-the-art unlearning method that utilizes approximation algorithms. This substantial improvement is evident as KNOT reaches target accuracies in a fraction of the time required by traditional methods—reducing training times from over 4400 s to just 660 s in specific scenarios.

6.6.8. HDUS

Definition

Paper [41] presents a technique called Heterogeneous Decentralized Unlearning with Seed Model Distillation (HDUS), designed for decentralized learning environments where clients may have different types of models. The technique addresses the process of handling

unlearning requests, where a client's contribution needs to be removed from the system without relying on a central server.

In HDUS, each client has a local dataset and a model trained on those data by minimizing a loss function. In addition to their main models, clients maintain seed models $\theta_{\text{seed},i}$, which are simplified versions of their main models. These seed models are distilled from the main models of neighboring clients using a shared reference dataset $X_{\text{ref},i}$. The distillation process aims to align the seed model with the main model by minimizing a distillation loss:

$$\mathcal{L}^* = \min_{\theta_{\text{seed},i}} \mathcal{L}(f(\theta_{\text{seed},i}, X_{\text{ref},i}), \sigma_T(f(\theta_i, X_{\text{ref},i}))) \tag{34}$$

where $\sigma_T(\cdot)$ is a temperature-scaled softmax function used to adjust the alignment process.

For inference, clients use an ensemble model that combines their own main model $f(\theta_i, \cdot)$ with the seed models $\{\theta_{\text{seed},j}\}$ from their neighbors:

$$F(f(\theta_i, x), S_i) = (1-\lambda) f(\theta_i, x) + \lambda \frac{1}{K} \sum_{k=1}^{K} f(\theta_{\text{seed},k}, x) \tag{35}$$

Here, λ is a parameter that controls the balance between the client's main model and the seed models from its neighbors. When a client leaves the network and requests that its contributions be unlearned, HDUS removes the corresponding seed model $\theta_{\text{seed},j}$ from its ensemble, effectively erasing the departing client's influence. The updated ensemble model is

$$F(f(\theta_i, x), S_i - \theta_{\text{seed},j}) = (1-\lambda) f(\theta_i, x) + \lambda \frac{1}{K-1} \sum_{k=1 k \neq j}^{K} f(\theta_{\text{seed},k}, x) \tag{36}$$

This process ensures that the unlearning is handled without the need to retrain the main models.

HDUS supports decentralized learning without a central server and is compatible with heterogeneous models across different clients. By using seed models, HDUS prevents the fusion of knowledge across the network, simplifying the unlearning process and making it more scalable for real-world applications.

Metric

The evaluation involves comparing HDUS to three baseline methods: Isolated stochastic gradient descent, shared incremental Sample-wise Unlearning, and Decentralized Stochastic Gradient Descent. The metrics used for evaluation include average test accuracy and standard deviation across multiple independent runs. HDUS shows better performance in heterogeneous model settings, where the models differ in architecture across clients. In the homogeneous setting, HDUS performs similarly to other distributed learning frameworks. In the heterogeneous setting, HDUS surpasses the baseline methods by maintaining a smaller accuracy drop—specifically, an average loss of 1.25% compared to larger drops in other methods.

6.6.9. Erase a Client

Definition

The proposed approach in paper [42] is divided into two main phases: local unlearning at the target client and federated learning post-training involving the server and remaining clients. In the first phase, local unlearning, the target client reverses its contribution to the global model by maximizing its local loss function, a process that contrasts with the typical minimization approach used during training. The local unlearning problem is

framed as a constrained optimization task. Specifically, the target client maximizes its local empirical risk:

$$F_i(w) = \frac{1}{n_i} \sum_{j \in D_i} L(w; (x_j, y_j)) \tag{37}$$

The optimization is constrained to an ℓ_2-norm ball around a reference model w_{ref}, ensuring that the model parameters remain close to what they would have been had the target client not participated in training. Mathematically, this is expressed as

$$\max_{w \in \{v \in \mathbb{R}^d : \|v - w_{\text{ref}}\|_2 \leq \delta\}} F_i(w) \tag{38}$$

where δ is a hyperparameter controlling the radius of the ℓ_2-norm ball. The reference model w_{ref} is computed as the average of the models from all other clients:

$$w_{\text{ref}} = \frac{1}{N-1} \sum_{j \neq i} w_{T-1}^j \tag{39}$$

where w_T is the global model after T rounds, w_{T-1}^i is the local model update from client i in round $T-1$, and N is the total number of clients. To solve this optimization problem, the paper employs Projected Gradient Descent (PGD), where the model parameters are iteratively updated and projected back onto the constrained region defined by the ℓ_2-norm ball. In the second phase, federated learning post-training, the server and remaining clients perform additional rounds of federated learning starting from the locally unlearned model w_u^i. This phase aims to fine-tune the model, ensuring that its performance on the retained clients' data is restored without the need for full retraining.

Metric

The paper evaluates the proposed federated unlearning technique by comparing its performance to the baseline of full model retraining, focusing on metrics such as efficacy, fidelity, and efficiency. In the scenario involving backdoor triggers, the proposed method reduces the backdoor accuracy to levels comparable to retraining. For example, in the MNIST dataset with $N = 5$ clients, the backdoor accuracy drops to around 0.02% using the proposed method, matching the 0.01% achieved by retraining. Similarly, for the flipped images scenario, the proposed method achieves a flipped accuracy of approximately 2.5% in the EMNIST dataset, compared to 2.1% with retraining, demonstrating comparable efficacy. Regarding efficiency, the proposed method shows a reduction in communication costs. In the MNIST dataset, achieving a clean accuracy of 98.13% requires 566 MB of communication with the proposed method, while retraining demands 1039 MB, making the proposed method 1.8 times more efficient.

6.6.10. QUICKDROP

Definition

QUICKDROP, proposed in [43], enhances federated unlearning by incorporating Dataset Distillation (DD) to minimize computational costs when removing specific data from a trained model. The essence of this approach lies in condensing the original dataset D into a much smaller synthetic dataset S, where $|S| \ll |D|$. This synthetic dataset S encapsulates the crucial information of D, making it feasible to use S for unlearning instead of the entire original dataset.

In the QUICKDROP framework, each client i independently distills its local dataset D_i into a corresponding smaller dataset S_i. The federated unlearning process is then restructured to operate on these distilled datasets. The unlearning operation is formally

defined as $\theta_f = U(\theta, S_f)$, where θ_f represents the unlearned model, θ is the original global model, and S_f is the distilled version of the dataset D_f that must be unlearned. The objective is to modify the model θ_f such that it no longer retains the knowledge from S_f while continuing to perform well on the remaining data $D \setminus D_f$.

QUICKDROP achieves this unlearning by employing stochastic gradient ascent (SGA) during the unlearning phase. In this phase, the model parameters are adjusted in a way that intentionally increases the loss on the target data S_f, thereby "forgetting" the learned information specific to that data. Mathematically, the update rule during the unlearning phase can be described as

$$\theta \leftarrow \theta + \eta \nabla_\theta \ell(\theta, S_f) \tag{40}$$

where η is the learning rate, and $\nabla_\theta \ell(\theta, S_f)$ is the gradient of the loss function ℓ with respect to the model parameters, calculated using the synthetic data S_f. This gradient ascent step effectively degrades the model's performance on the data represented by S_f.

After the unlearning phase, the model undergoes a recovery phase to restore its performance on the remaining data $D \setminus D_f$. During this recovery, the model is retrained using the distilled dataset S combined with a small portion of the original data, ensuring that the model regains its accuracy on the non-target data.

Metric

QUICKDROP's performance was evaluated using metrics such as testing accuracy on the forgotten set (F-Set) and the remaining set (R-Set) after the unlearning and recovery stages. In single-class unlearning scenarios, QUICKDROP effectively reduced the F-Set accuracy to 0.85% post-unlearning, demonstrating the technique's ability to remove the target class information from the model. After the recovery phase, QUICKDROP achieved an R-Set accuracy of 71.98%, slightly lower than baselines like stochastic gradient ascent on the original dataset (SGA-OR), Federated Unlearning using Model Pruning (FU-MP), and retraining the model from scratch (RETRAIN-OR), which had R-Set accuracies ranging from 72.83% to 73.45%. When tested under sequential unlearning requests, QUICKDROP consistently reduced the target class accuracy to near zero while restoring the non-target class accuracy during the recovery phase. Additionally, QUICKDROP exhibited a substantial improvement in computational efficiency, executing unlearning requests 475.2 times faster than the retraining-based approach.

7. Discussion of Results

In this section, a discussion on machine unlearning techniques is presented, focusing on their need for fine-tuning and the level of unlearning they achieve. The techniques vary in their reliance on fine-tuning after modifying data to maintain model performance, and the analysis examines the prevalence of instance-level versus class-level unlearning strategies. Instance-level unlearning removes individual data points as requested, while class-level unlearning allows broader modifications by removing entire data clusters at once. Key insights and observations are also highlighted.

7.1. Data Based

The predominance of data-based machine unlearning techniques targeting single or multiple class levels, as observed in Table 8, can be attributed to the relative simplicity and efficiency of creating and managing patterns for unlearning entire classes. Creating and maintaining patterns for unlearning entire classes is less resource-intensive than handling individual instances. For example, using a trojan trigger or mnemonic code for a whole class involves maintaining a single pattern per class, rather than a unique pattern for each data point. This significantly reduces the overhead in terms of storage and

computational complexity, making it easier to implement and maintain. Consequently, the higher percentage of techniques focusing on class-level unlearning is a natural outcome of these efficiencies.

Table 8. Fine-tuning requirements and levels of unlearning of data-based techniques.

ID	Is Fine-Tuning or Retraining Necessary?	Notes
1 [2,+]	✓	Specific details such as the number of epochs or the amount of data required for retraining after applying the trojan trigger were not explicitly mentioned.
2 [2,+]	✓	Requires incremental retraining with a few epochs and a small amount of data after applying redaction.
3 [2,+]	✓	Requires a repair step involving fine-tuning on a subset of the original retain set to restore accuracy.
4 [1,+]	✓	Fine-tuning may suffice for minor adjustments without substantial model reconfiguration.

[1] Single or multiple class level. [2] Single instance level. [+] Refer to Table 3 for information about the primary study.

7.2. Architecture Based

7.2.1. Modular Unlearning

Analyzing the various techniques reveals a common thread: the pursuit of minimizing retraining efforts, as shown in Table 9. Each iteration in modular unlearning techniques builds upon previous advancements, striving to streamline the process of updating models after data removal. This collective endeavor tries to maintain model accuracy and efficiency within evolving data landscapes. Techniques like those employing differential privacy or encoded data representations exemplify this trend, aiming to reduce computational overhead and preserve model integrity without compromising performance. The evolution towards more efficient retraining strategies underscores the field's maturation, reflecting ongoing efforts to operationalize machine unlearning in real-world applications. Furthermore, the predominance of instance-level unlearning techniques in modular architectures can be attributed to several factors rooted in their architectural design and operational requirements. This architectural granularity allows for targeted updates and adjustments at the level of individual instances within these partitions, as observed in Table 9. This approach underscores a deliberate effort to refine model adaptations precisely where necessary, optimizing performance without overhauling entire datasets.

An observation from these advancements is the prevalence of SISA in the literature on machine unlearning. Most reviewed papers (21/37) reference SISA in their related work or comparative analyses, highlighting its foundational role. This widespread citation underscores SISA's influence as a benchmark for evaluating new methodologies and innovations in adaptive machine learning systems. By creating consistent methods for dealing with unlearning requests, SISA has driven progress in making models more

adaptable and reducing privacy risks in changing data environments. Its enduring presence in scholarly discourse underscores its pivotal role in shaping the trajectory of modular unlearning research.

Table 9. Fine-tuning requirements and levels of unlearning of modular techniques.

ID	Is Fine-Tuning or Retraining Necessary?	Notes
5 [2,+]	✓	Details regarding the extent and methodology of fine-tuning required were not explicitly mentioned.
6 [2,+]	✓	Requires fine-tuning with a specific subset of data to address the adaptive nature of the unlearning process.
7 [2,+]	✓	Fine-tuning necessary to maintain performance consistency after implementing unlearning techniques.
8 [2,+]	✓	Incremental retraining required to ensure the integrity and accuracy of the model post-unlearning.
9 [1,+]	✓	-
10 [1,+]	✓	-

[1] Single or multiple class level. [2] Single instance level. [+] Refer to Table 3 for information about the primary study.

7.2.2. Gradient Ascent

The gradient ascent methods require meticulous tracking of each training batch's contribution during model training. The storage requirements for each batch are proportional to the model size and number of training steps. When a batch containing sensitive or unwanted data is identified for removal, the unlearning process typically involves subtracting the accumulated parameter updates associated with those data points from the final model parameters. However, this efficiency comes with its own set of challenges and trade-offs. Storing the indices of examples participating in each batch and their corresponding updates requires considerable storage capacity. This can be significant, particularly for large models and/or extensive training runs [17]. Additionally, this method might cause the model to be different from what it would have been if those updates were never made, especially with larger datasets and more complicated training processes.

Another observation from the literature of this type of technique is the predominant focus on forgetting at the instance level rather than at the level of entire classes (Table 10). There is an absence of methods explicitly designed to forget entire data classes. This is because these techniques unlearn one data point at a time, requiring meticulous tracking and recording of each training batch's contributions during the model training process. Consequently, as these techniques adjust model parameters based on specific instances, they experience a gradual loss of accuracy or performance with each new request for unlearning.

Given the existing research, retraining may not be immediately necessary when the unlearning involves straightforward adjustments to model parameters or for simpler models (Table 10). However, in more complex models such as deep neural networks, where

parameters are highly interconnected and changes to individual data points can have ripple effects across the model, fine-tuning or retraining prove to be beneficial. This ensures that the model adapts to the new data distribution post-unlearning and maintains or improves performance on unseen data.

Table 10. Fine-tuning requirements and levels of unlearning of gradient ascent techniques.

ID	Is Fine-Tuning or Retraining Necessary?	Notes
11 [2,+]	✓	Some retraining is usually performed afterward to restore model performance on non-target data.
12 [2,+]	x	-
13 [2,+]	✓	-
14 [2,+]	✓	It needs some epochs of training using clean data and the identified trigger patterns.

[2] Single instance level. [+] Refer to Table 3 for information about the primary study.

7.2.3. Teacher–Student

All the techniques involving the teacher–student framework inherently perform a form of fine-tuning due to their operational methodology, as evidenced in Table 11. These techniques utilize an iterative process, where the student model is progressively adjusted based on the guidance provided by the teacher model. This approach mirrors fine-tuning, where the student model undergoes incremental updates to align with the teacher's outputs and to unlearn specific data. The iterative nature of these adjustments ensures that the student model refines its performance continually, similar to how fine-tuning hones a pretrained model for specific tasks.

Table 11. Fine-tuning requirements and levels of unlearning of teacher–student techniques.

ID	Is Fine-Tuning or Retraining Necessary?
15 [1,+]	✓
16 [2,+]	✓
17 [1,+]	✓
18 [2,+]	✓
19 [1,+]	✓
20 [2,+]	✓

[1] Single instance level. [2] Single or multiple class level. [+] Refer to Table 3 for information about the primary study.

An insight from examining these techniques is the balanced distribution at the scope of unlearning, showing an equal split between single/multiple class-level and single/multiple instance-level unlearning, as shown in Table 11. Techniques like bad teaching, where the student learns from both competent and incompetent teachers, enable class-level unlearning through generalized learning objectives. Other methods involving generative adversarial networks and gated knowledge transfer use pseudo samples to facilitate selective unlearning and ensure the student model forgets targeted information.

While reviewing the techniques described in the literature, it becomes evident that many rely heavily on distance functions to guide the process of unlearning. By leveraging distance functions, these methods aim to minimize the discrepancy between the original model and the adjusted model post-learning. This ensures that retained knowledge remains intact while forgotten information is effectively erased or modified. However, the choice and design of these distance functions are pivotal, as they directly influence the effectiveness and efficiency of the unlearning process. Inappropriate or overly complex distance metrics may introduce unnecessary computational overhead or obscure insights into model behavior.

The Student and Teacher Framework section reveals a prevalent trend towards iterative methodologies in machine unlearning techniques. The iterative nature ensures adaptability to dynamic datasets but also highlights a limitation: computational overhead due to repeated model adjustments. This iterative requirement suggests that while effectively managing targeted forgetting, these techniques may demand substantial computational resources, potentially limiting their scalability in real-time or resource-constrained environments. Upon reviewing the literature, it is also evident that another downside of these techniques is the reliance on maintaining more than one model simultaneously, which can be costly in terms of computational resources and storage.

7.2.4. Scrubbing Weights Approach

The scrubbing weights approach encompasses a diverse array of techniques designed to effectively eliminate the influence of specific data points from machine learning models. These methods leverage sophisticated mathematical frameworks such as Hessians and Fisher information matrices to ensure precise data removal grounded in theory. Incorporating controlled noise into model weights is also a commonly adopted strategy that ensures efficient forgetting without compromising model performance. To tackle computational complexities in handling large and intricate models, some approaches utilize influence functions and approximate Hessian matrices, ensuring scalability and practical feasibility.

However, while these techniques offer robust solutions, they also present inherent limitations. Despite avoiding full retraining, they often require intricate mathematical computations and approximations, such as calculating Fisher information matrices or approximating Hessians, which can introduce computational overhead. Some methods rely on linear approximations or synthetic predictions, which may not fully capture the complexities of nonlinear models, potentially leading to suboptimal forgetting outcomes. Successful implementation hinges on accurate parameter estimation and transformation, with errors in these approximations posing risks to the effectiveness of data removal. Moreover, while efforts are made to minimize residual information, challenges persist in ensuring complete data erasure and preventing information leakage.

A strength identified in the literature is that most of these techniques do not necessitate retraining or fine-tuning, as indicated in Table 12. Instead, they directly adjust model parameters based on calculated modifications that counteract the influence of targeted data points. This characteristic allows these methods to operate as single-step post-processing procedures, significantly reducing computational time and resource requirements compared to iterative retraining approaches. One method [31] deviates by iteratively adjusting model weights, resembling retraining processes to a certain extent, unlike the straightforward approach of other techniques.

All these methods primarily focus on multiple instance unlearning. This is due to their reliance on mathematical frameworks designed to handle aggregate data contributions, making it more efficient to process multiple instances simultaneously rather than focusing on individual data points.

Table 12. Fine-tuning requirements and levels of unlearning of scrubbing weights techniques.

ID	Is Fine-Tuning or Retraining Necessary?
21 [2,+]	x
22 [2,+]	x
23 [2,+]	x
24 [2,+]	x
25 [2,+]	✓
26 [2,+]	x
27 [2,+]	x

[2] Single or multiple class level. [+] Refer to Table 3 for information about the primary study.

7.2.5. Federated Unlearning

Table 13 shows that most unlearning techniques require some form of retraining or fine-tuning, despite differences in unlearning granularity. Erasing specific knowledge from a model often creates gaps or imbalances in its decision-making process. Fine-tuning is essential to recalibrate the model and ensure it functions correctly without the removed data. This is especially important in federated learning, where models must generalize across diverse and decentralized data sources.

Table 13. Fine-tuning requirements and levels of unlearning of federated unlearning techniques.

ID	Is Fine-Tuning or Retraining Necessary?	Notes
28 [2,+]	x	-
29 [2,+]	✓	Fine-tuning process to integrate the distilled knowledge effectively
30 [1,+]	✓	The pruned model is retrained using the non-target categories.
31 [2,+]	✓	Adjust the model parameters efficiently based on the FIM updates
32 [2,+]	x	-
33 [2,+]	✓	It consists of iteratively applying restricted gradient ascent
34 [2,+]	✓	-
35 [2,+]	x	-
36 [2,+]	✓	-
37 [1,+]	✓	Fine-tuned using the distilled datasets from the non-target classes

[1] Single or multiple class level. [2] Client level. [+] Refer to Table 3 for information about the primary study.

In federated unlearning, most techniques focus on client-level contributions, with fewer methods addressing class-level unlearning and even fewer targeting specific instances. Client-level unlearning is prioritized because federated learning aggregates up-

dates from multiple clients. Removing the influence of a particular client or group of clients is more straightforward and aligns with the federated structure. Class-level unlearning involves identifying and removing contributions related to specific classes across all clients. The limited use of instance-level unlearning in federated settings is likely due to its complexity and computational demands. Instance-level unlearning requires identifying and removing the influence of specific data points, which is challenging in a federated system with decentralized data.

7.3. Dataset

Analyzing dataset usage across different machine unlearning techniques provides insights into research trends and methodological choices within the field, as illustrated in Figure 6. CIFAR-10 emerges as a predominant choice across various techniques, reflecting its suitability for evaluating unlearning methods in complex image classification tasks. Its diverse range of objects and scenes allows researchers to assess model adaptability and robustness across different categories, ensuring a comprehensive evaluation of technique efficacy. MNIST, renowned for its simplicity and well-defined character recognition task, is frequently utilized in studies focusing on modular unlearning and scrubbing weights approaches. This dataset facilitates the evaluation of unlearning effects on basic classification tasks, providing insights into model behavior post-unlearning.

In contrast, specialized datasets such as HAM10000 and VGG-Faces feature prominently in teacher–student approaches, chosen for their relevance to specific applications like dermatology and facial recognition. These datasets enable researchers to evaluate unlearning techniques in contexts requiring nuanced model adjustments and fine-grained knowledge transfer between models. Imagenet, though less frequently used, appears in studies exploring modular unlearning and scrubbing weight approaches, leveraging its image diversity to assess technique performance in broader, more complex visual recognition tasks.

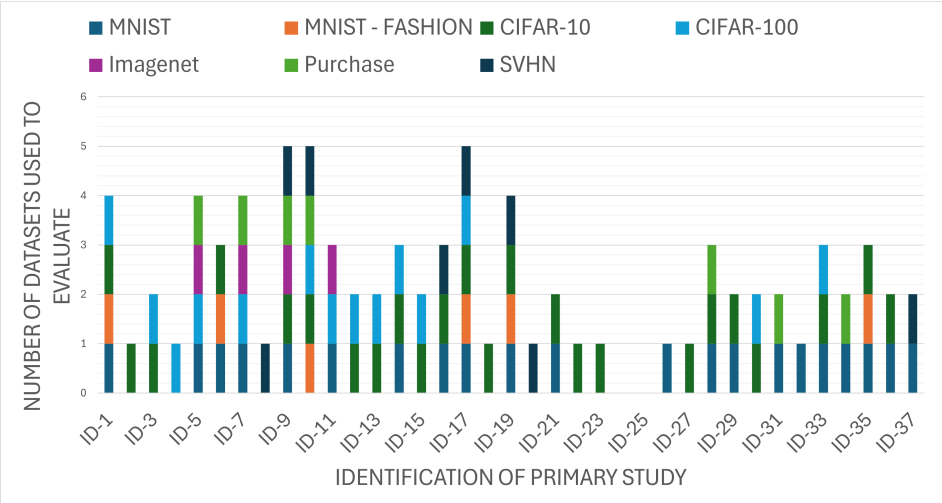

Figure 6. Distribution of datasets utilized across different machine unlearning techniques.

Papers like [30,31], which do not employ any datasets in their evaluations, indicate a focus on theoretical validation. This trend suggests a dual approach in machine unlearning research: while leveraging established benchmark datasets for generalizable insights, researchers also explore domain-specific datasets for task-specific evaluations. Moreover, almost all the datasets used across the analyzed papers are primarily intended for classification tasks. However, notable exceptions such as AgeDB, as observed in [26], demonstrate a unique suitability for regression tasks due to its annotation of age attributes for various

subjects, encompassing a wide range of ages and identities. This dataset's utilization underscores the versatility of machine unlearning techniques beyond classification, extending into domains requiring predictions of continuous outcome variables like age.

The analysis reveals a predominant use of convolutional neural networks in machine unlearning research. CNNs are used due to their effectiveness in image classification tasks and ability to capture spatial hierarchies through convolutional layers. Despite the focus on CNNs, other neural network architectures, such as Recurrent Neural Networks and Long Short-Term Memory networks, could also be considered. These architectures are particularly useful for sequential data and time-series analysis, which opens possibilities for machine unlearning applications beyond image classification. Incorporating these neural network types could broaden the scope of machine unlearning research and provide insights into the unlearning process for different data modalities.

7.4. Architecture

The analysis of architectures used in various techniques reveals several trends and preferences, as depicted in Figure 7. ResNet, a family of convolutional neural networks, is extensively used across the studies. ResNet architectures, including ResNet-18, ResNet-50, and ResNet-20, are known for their residual learning framework, which addresses the vanishing gradient problem by allowing gradients to flow through the network via shortcut connections [60]. The numbers in these architectures (18, 50, 20) refer to the depth of the network—specifically, the number of layers. The increased depth in architectures like ResNet-50 allows for more complex feature extraction, while the residual connections help maintain the flow of gradients, thus facilitating the training of very deep networks. ResNet's robustness and versatility make it suitable for a wide range of machine unlearning evaluations. Another common architecture is the VGG family, particularly VGG-16, used in four papers. VGG-16 [61] employs a stack of convolutional layers with small receptive fields (3×3 filters) followed by fully connected layers. The number 16 in VGG-16 denotes the total number of layers in the network. VGG-16's design, with its consistent layer structure and uniform filter size, makes it computationally efficient and straightforward to implement. This simplicity is an advantage for benchmarking in machine unlearning research, as it allows for clear comparisons of model performance. The use of VGG-16 suggests a trend towards leveraging established architectures recognized for their performance in various computer vision tasks.

Including MobileNetv2 [62] in two studies indicates an interest in efficient and lightweight models. MobileNetv2 is designed for mobile and embedded vision applications, suggesting that researchers are considering the implications of deploying machine unlearning techniques in resource-constrained environments. Additionally, the use of DenseNet [63] underscores the importance of architectures that enhance feature reuse and reduce the number of parameters. DenseNet's dense connectivity pattern helps mitigate the vanishing gradient problem and improves information flow, making it a choice for machine unlearning evaluations.

There is also diversity in architecture choices, with some studies employing multiple architectures to evaluate their techniques comprehensively. For example, one study [9] uses ResNet-18, All-CNN, and MobileNetv2, while another explores LeNet, ResNet, and VGG. This approach indicates a trend towards thorough benchmarking across different model complexities and capacities, ensuring that the proposed unlearning techniques are robust and generalizable. Furthermore, the analysis also reveals that two papers do not specify any architectures, focusing instead on linear models and generalizing their findings to deep neural networks. These studies concentrate on theoretical validation, developing foundational principles that can be applied broadly. Despite generalizing their results to

neural networks, these papers do not conduct specific experiments or evaluations involving particular neural network architectures.

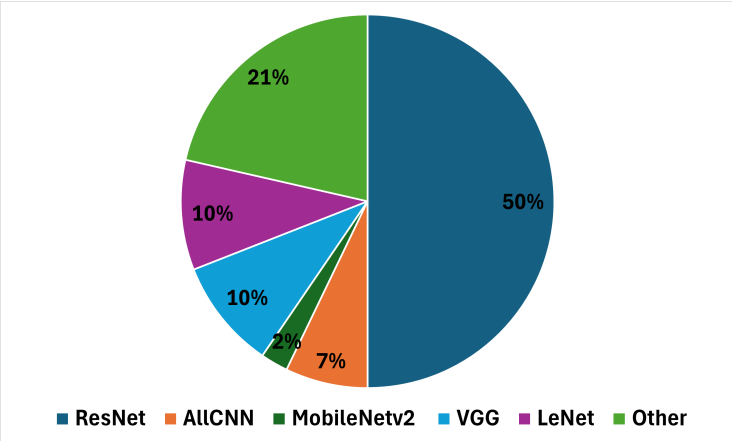

Figure 7. Overview of neural network architectures utilized in machine unlearning technique evaluations.

7.5. Replicability

The availability of source code or pseudocode significantly influences the replicability of machine unlearning techniques. Techniques where authors provide comprehensive source code facilitate easier replication by allowing other researchers to implement and verify the methods described directly. Pseudocode also plays a positive role in replicability. While it requires more interpretation than executable code, it provides a structured outline of the algorithmic steps involved. However, a notable portion of the techniques reviewed in the table do not provide either source code or pseudocode. This absence poses challenges to replication efforts, as researchers must rely solely on the methodological descriptions provided in the papers. Replicating these studies becomes more time-consuming and prone to interpretation errors, potentially leading to variations in results. The distribution of papers with pseudocode, source code repositories, and those without is illustrated in Figure 8.

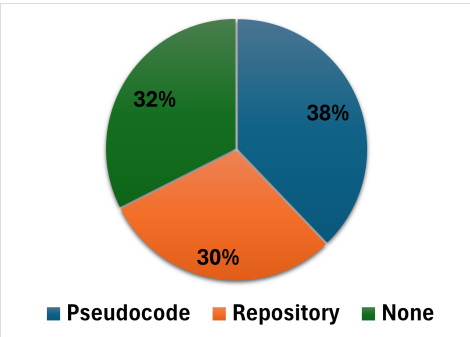

Figure 8. Distribution of machine unlearning papers by the availability of code repositories, pseudocode, or neither.

Figure 9 highlights a diverse range of metrics used across different studies to evaluate machine unlearning techniques. This variability suggests that there is no universal standard or consensus on which metrics are most appropriate for assessing the effectiveness of unlearning approaches. Many studies prioritize metrics related to privacy and security, such as membership inference and model inversion attacks. These metrics are crucial for determining the extent to which unlearned models retain sensitive information and their vulnerability to privacy attacks. Metrics like accuracy on a forgotten set and relearn time

indicate studies' interest in understanding how unlearning techniques affect model performance. This consideration is essential for balancing privacy preservation and maintaining model effectiveness on retained data. Metrics such as unlearn time and activation distance reflect studies' concerns about the computational efficiency of unlearning techniques. Techniques that require less computational resources for unlearning are more practical for deployment in real-world applications.

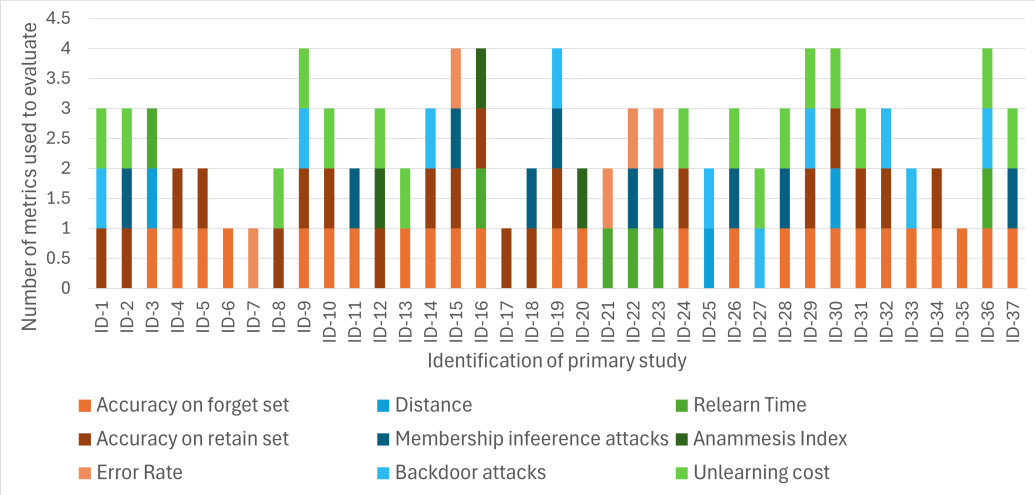

Figure 9. Distribution of metrics utilized across different machine unlearning techniques.

7.6. Real-World Applications

Machine unlearning has significant real-world applications across various industries where data privacy, compliance, and security are critical. This section highlights three key domains where unlearning techniques play a pivotal role: healthcare, finance, and facial recognition systems.

7.6.1. Healthcare

Patient data privacy and compliance with regulations such as HIPAA and GDPR necessitate the ability to erase sensitive records efficiently. Machine unlearning techniques ensure that when a patient revokes consent for data usage, their medical history can be removed from AI models without requiring a complete retraining process [64]. This capability is crucial for preventing the unauthorized use of patient data in predictive models for disease diagnosis, treatment planning, and medical imaging systems [22].

7.6.2. Finance

Financial institutions use AI-driven models for credit scoring, fraud detection, and risk assessment. However, erroneous or outdated information in these models can lead to biased decision-making, potentially affecting users unfairly. Machine unlearning allows financial organizations to update and remove customer data as needed, enhancing compliance with data protection laws and ensuring fairness in decision-making processes [65]. Additionally, unlearning methods improve robustness against adversarial attacks targeting financial models [9].

7.6.3. Facial Recognition

Facial recognition systems rely on vast datasets for training, often containing personally identifiable information. Machine unlearning enables compliance with privacy regulations by ensuring that individuals who opt out of such datasets have their data erased without residual traces in the model [66]. This is particularly relevant for applications in surveillance, law enforcement, and biometric authentication, where data protection is

a growing concern. Furthermore, unlearning helps mitigate biases in facial recognition models by allowing targeted removal of biased training samples [67].

The integration of unlearning techniques into these industries marks a shift towards more ethical and privacy-preserving AI applications, aligning with global data protection mandates and user expectations.

7.7. Final Comparison of Machine Unlearning Approaches

To better understand the trade-offs between various machine unlearning strategies, their key characteristics are summarized and shown in Table 14, focusing on their applicability and limitations across different scenarios.

Table 14. Advantages, disadvantages, and use cases for machine unlearning categories.

Category	Advantages	Disadvantages	Use Cases
Data-Based	- Simplicity and straightforward implementation. - Minimal model architecture changes. - Efficient for small datasets.	- Performance impact for large-scale data. - May require retraining or fine-tuning. - Limited scalability for complex models.	- Individual data point or class removal. - Privacy-sensitive applications with manageable datasets.
Architecture-Based			
Modular Unlearning	- Focused removal of specific modules. - Minimal impact on unrelated model components.	- Requires modular design. - Not suitable for monolithic architectures.	- Models with modular or layered structures (e.g., multi-task models).
Gradient Ascent	- Directly modifies weights to undo learning effects. - Handles selective unlearning efficiently.	- High computational cost. - Risk of overfitting during weight updates.	- Removing specific data points or subsets in non-convex models.
Teacher–Student	- Ensures comprehensive unlearning via model distillation. - Provides provable forgetting guarantees.	- Requires training a new "student" model. - Resource-intensive for large datasets or models.	- Scenarios requiring high assurance of unlearning (e.g., regulatory compliance).
Scrubbing Weight Approach	- Targets specific information encoded in model weights. - Preserves model utility for unaffected data.	- Requires techniques to identify and scrub relevant weights. - May not guarantee full forgetting.	- Handling sensitive data encoded in specific parts of the model.
Federated Unlearning	- Decentralized approach reduces central computation. - Can selectively unlearn client-specific data.	- Complex implementation. - Limited ability to unlearn individual data points from a client.	- Applications involving distributed datasets, such as federated learning in healthcare or finance.

While the aforementioned methods offer effective solutions for various unlearning scenarios, scalability remains a critical challenge, particularly for large-scale neural networks. Many current techniques, such as gradient ascent unlearning and teacher–student model distillation, require intensive computational resources and are often impractical for large-scale architectures due to their iterative nature [68]. A promising direction to address scalability concerns is leveraging modular and layer-wise unlearning approaches, which limit modifications to localized network components, significantly reducing com-

putational overhead [69]. Additionally, advances in federated unlearning distribute the computational burden across decentralized nodes, allowing for more scalable solutions in privacy-sensitive applications while maintaining performance efficiency [66]. Future work should focus on refining scalable unlearning techniques that preserve model integrity without the need for full retraining, enabling their deployment in real-world, large-scale AI systems.

8. Conclusions and Future Work

This review examined the field of machine unlearning driven by data privacy regulations like GDPR and CCPA. By analyzing 37 selected primary studies, it thoroughly evaluated the foundational principles, key performance metrics, and methodologies used to assess these techniques in both regression and classification tasks. Special attention was given to recent advancements, categorizing and detailing unlearning techniques to offer deeper insights into their evolution, effectiveness, efficiency, and broader applicability. This work aimed to provide a solid foundation for future research, development, and practical implementations in data privacy, model management, and compliance.

The challenges of selectively removing data contributions at both client and instance levels were discussed, highlighting the balance between computational costs and privacy guarantees. The analysis of various data-based, architecture-based, and federated unlearning techniques revealed different approaches and their associated fine-tuning requirements and levels of unlearning achieved. The prevalence of SISA in the literature was noted, highlighting its foundational role as a benchmark.

We observed the persistent need for fine-tuning in many unlearning techniques and the varied levels of unlearning they achieve, with a predominant focus on instance-level forgetting. The reliance on distance functions in guiding the unlearning process was also observed. A critical challenge identified across various methods is scalability, especially for large-scale neural networks.

A prominent area of future work is refining scalable unlearning techniques that can effectively remove specific data while preserving the integrity and utility of the model on the remaining data. Further research could also address the lack of methods explicitly designed to forget entire data classes. The variability in evaluation metrics across different studies also points to a need for a more unified framework for evaluating unlearning methods. Moreover, exploring the applicability of machine unlearning to a broader range of neural network architectures beyond the commonly used CNNs and MLPs could be a valuable direction for future investigation.

Author Contributions: Conceptualization, validation, supervision, project administration, and funding acquisition: M.E.B.; methodology, software, investigation, data curation, and writing—original draft preparation: I.D.C. and J.A.Z.; formal analysis: M.E.B., Á.L.V.C., and L.I.B.-L., writing—review and editing: Á.L.V.C. and L.I.B.-L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data is contained within the article.

Acknowledgments: The authors would like to thank the Faculty of Systems Engineering, Escuela Politécnica Nacional, Quito 170517, Ecuador for academic and financial support.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Art. 17 GDPR—Right to Erasure ('Right to Be Forgotten')—GDPR.eu—Gdpr.eu. 2018. Available online: https://gdpr.eu/article-17-right-to-be-forgotten/ (accessed on 13 July 2024).
2. California Consumer Privacy Act (CCPA)—Oag.ca.gov. 2024. Available online: https://oag.ca.gov/privacy/ccpa#heading5d (accessed on 13 July 2024).
3. Nast, C. Now That Machines Can Learn, Can They Unlearn?—Wired.com. 2021. Available online: https://www.wired.com/story/machines-can-learn-can-they-unlearn/ (accessed on 13 July 2024).
4. Shaik, T.; Tao, X.; Xie, H.; Li, L.; Zhu, X.; Li, Q. Exploring the Landscape of Machine Unlearning: A Comprehensive Survey and Taxonomy. *arXiv* **2024**, arXiv:2305.06360.
5. Li, C. OpenAI's GPT-3 Language Model: A Technical Overview—Lambdalabs.com. 2020. Available online: https://lambdalabs.com/blog/demystifying-gpt-3 (accessed on 13 July 2024).
6. Kitchenham, B. Procedures for performing systematic reviews. *Keele Uk Keele Univ.* **2004**, *33*, 1–26.
7. Liu, Y.; Ma, S.; Aafer, Y.; Lee, W.C.; Zhai, J.; Wang, W.; Zhang, X. Trojaning Attack on Neural Networks. In Proceedings of the Network and Distributed System Security Symposium, San Diego, CA, USA, 18–21 February 2018.
8. Felps, D.L.; Schwickerath, A.D.; Williams, J.D.; Vuong, T.N.; Briggs, A.; Hunt, M.; Sakmar, E.; Saranchak, D.D.; Shumaker, T. Class Clown: Data Redaction in Machine Unlearning at Enterprise Scale. *arXiv* **2020**, arXiv:/2012.04699.
9. Tarun, A.K.; Chundawat, V.S.; Mandal, M.; Kankanhalli, M. Fast Yet Effective Machine Unlearning. *IEEE Trans. Neural Netw. Learn. Syst.* **2024**, *35*, 13046–13055. [CrossRef]
10. Shibata, T.; Irie, G.; Ikami, D.; Mitsuzumi, Y. Learning with Selective Forgetting. In Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21; International Joint Conferences on Artificial Intelligence Organization, Montreal, QC, Canada, 19–27 August 2021; Zhou, Z.H., Ed.; pp. 989–996. [CrossRef]
11. Bourtoule, L.; Chandrasekaran, V.; Choquette-Choo, C.A.; Jia, H.; Travers, A.; Zhang, B.; Lie, D.; Papernot, N. Machine Unlearning. *arXiv* **2020**, arXiv:1912.03817.
12. Gupta, V.; Jung, C.; Neel, S.; Roth, A.; Sharifi-Malvajerdi, S.; Waites, C. Adaptive Machine Unlearning. *arXiv* **2021**, arXiv:2106.04378.
13. Koch, K.; Soll, M. No Matter How You Slice It: Machine Unlearning with SISA Comes at the Expense of Minority Classes. In Proceedings of the 2023 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML), Raleigh, NC, USA, 8–10 February 2023.
14. Aldaghri, N.; Mahdavifar, H.; Beirami, A. Coded Machine Unlearning. *IEEE Access* **2021**, *9*, 88137–88150. [CrossRef]
15. He, Y.; Meng, G.; Chen, K.; He, J.; Hu, X. DeepObliviate: A Powerful Charm for Erasing Data Residual Memory in Deep Neural Networks. *arXiv* **2021**, arXiv:2105.06209.
16. Yan, H.; Li, X.; Guo, Z.; Li, H.; Li, F.; Lin, X. ARCANE: An Efficient Architecture for Exact Machine Unlearning. In Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22; International Joint Conferences on Artificial Intelligence Organization, Vienna, Austria, 23–29 July 2022; Raedt, L.D., Ed.; pp. 4006–4013. [CrossRef]
17. Graves, L.; Nagisetty, V.; Ganesh, V. Amnesiac Machine Learning. *arXiv* **2020**, arXiv:2010.10981. [CrossRef]
18. Thudi, A.; Deza, G.; Chandrasekaran, V.; Papernot, N. Unrolling SGD: Understanding Factors Influencing Machine Unlearning. *arXiv* **2022**, arXiv:2109.13398.
19. Liu, Y.; Ma, Z.; Liu, X.; Liu, J.; Jiang, Z.; Ma, J.; Yu, P.; Ren, K. Learn to Forget: Machine Unlearning via Neuron Masking. *arXiv* **2021**, arXiv:2003.10933.
20. Liu, Y.; Fan, M.; Chen, C.; Liu, X.; Ma, Z.; Wang, L.; Ma, J. Backdoor Defense with Machine Unlearning. *arXiv* **2022**, arXiv:2201.09538.
21. Chundawat, V.S.; Tarun, A.K.; Mandal, M.; Kankanhalli, M. Can Bad Teaching Induce Forgetting? Unlearning in Deep Networks using an Incompetent Teacher. *arXiv* **2023**, arXiv:2205.08096.
22. Chundawat, V.S.; Tarun, A.K.; Mandal, M.; Kankanhalli, M. Zero-Shot Machine Unlearning. *IEEE Trans. Inf. Forensics Secur.* **2023**, *18*, 2345–2354. [CrossRef]
23. Kim, J.; Woo, S.S. Efficient Two-stage Model Retraining for Machine Unlearning. In Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), New Orleans, LA, USA, 19–20 June 2022; pp. 4360–4368. [CrossRef]
24. Kurmanji, M.; Triantafillou, P.; Hayes, J.; Triantafillou, E. Towards Unbounded Machine Unlearning. *arXiv* **2023**, arXiv:2302.09880.
25. Chen, K.; Wang, Y.; Huang, Y. Lightweight machine unlearning in neural network. *arXiv* **2021**, arXiv:2111.05528.
26. Tarun, A.K.; Chundawat, V.S.; Mandal, M.; Kankanhalli, M. Deep Regression Unlearning. *arXiv* **2023**, arXiv:2210.08196.
27. Golatkar, A.; Achille, A.; Soatto, S. Eternal Sunshine of the Spotless Net: Selective Forgetting in Deep Networks. *arXiv* **2020**, arXiv:1911.04933.
28. Golatkar, A.; Achille, A.; Soatto, S. Forgetting Outside the Box: Scrubbing Deep Networks of Information Accessible from Input-Output Observations. *arXiv* **2020**, arXiv:2003.02960.

29. Golatkar, A.; Achille, A.; Ravichandran, A.; Polito, M.; Soatto, S. Mixed-Privacy Forgetting in Deep Networks. *arXiv* **2021**, arXiv:2012.13431.
30. Guo, C.; Goldstein, T.; Hannun, A.; van der Maaten, L. Certified Data Removal from Machine Learning Models. *arXiv* **2023**, arXiv:1911.03030.
31. Izzo, Z.; Smart, M.A.; Chaudhuri, K.; Zou, J. Approximate Data Deletion from Machine Learning Models. *arXiv* **2021**, arXiv:2002.10077.
32. Wu, G.; Hashemi, M.; Srinivasa, C. PUMA: Performance Unchanged Model Augmentation for Training Data Removal. *arXiv* **2022**, arXiv:2203.00846. [CrossRef]
33. Warnecke, A.; Pirch, L.; Wressnegger, C.; Rieck, K. Machine Unlearning of Features and Labels. *arXiv* **2023**, arXiv:2108.11577.
34. Liu, G.; Ma, X.; Yang, Y.; Wang, C.; Liu, J. FedEraser: Enabling Efficient Client-Level Data Removal from Federated Learning Models. In Proceedings of the 2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS), Tokyo, Japan, 25–28 June 2021; pp. 1–10. [CrossRef]
35. Wu, C.; Zhu, S.; Mitra, P. Federated Unlearning with Knowledge Distillation. *arXiv* **2022**, arXiv:2201.09441.
36. Wang, J.; Guo, S.; Xie, X.; Qi, H. Federated Unlearning via Class-Discriminative Pruning. *arXiv* **2022**, arXiv:2110.11794.
37. Liu, Y.; Xu, L.; Yuan, X.; Wang, C.; Li, B. The Right to be Forgotten in Federated Learning: An Efficient Realization with Rapid Retraining. In Proceedings of the IEEE INFOCOM 2022 - IEEE Conference on Computer Communications, London, UK, 2–5 May 2022. [CrossRef]
38. Cao, X.; Jia, J.; Zhang, Z.; Gong, N.Z. FedRecover: Recovering from Poisoning Attacks in Federated Learning using Historical Information. *arXiv* **2022**, arXiv:2210.10936.
39. Li, G.; Shen, L.; Sun, Y.; Hu, Y.; Hu, H.; Tao, D. Subspace based Federated Unlearning. *arXiv* **2023**, arXiv:2302.12448.
40. Su, N.; Li, B. Asynchronous Federated Unlearning. *arXiv* **2023**, arXiv:2207.05521.
41. Ye, G.; Chen, T.; Nguyen, Q.V.H.; Yin, H. Heterogeneous Decentralized Machine Unlearning with Seed Model Distillation. *arXiv* **2023**, arXiv:2308.13269.
42. Halimi, A.; Kadhe, S.; Rawat, A.; Baracaldo, N. Federated Unlearning: How to Efficiently Erase a Client in FL? *arXiv* **2023**, arXiv:2207.05521.
43. Dhasade, A.; Ding, Y.; Guo, S.; marie Kermarrec, A.; Vos, M.D.; Wu, L. QuickDrop: Efficient Federated Unlearning by Integrated Dataset Distillation. *arXiv* **2023**, arXiv:2311.15603.
44. Cao, Y.; Yang, J. Towards Making Systems Forget with Machine Unlearning. In Proceedings of the 2015 IEEE Symposium on Security and Privacy, San Jose, CA, USA, 17–21 May 2015; pp. 463–480. [CrossRef]
45. Sekhari, A.; Acharya, J.; Kamath, G.; Suresh, A.T. Remember What You Want to Forget: Algorithms for Machine Unlearning. *arXiv* **2021**, arXiv:2103.03279.
46. Dwork, C.; Roth, A. The algorithmic foundations of differential privacy. *Found. Trends® Theor. Comput. Sci.* **2013**, *9*, 211–407. [CrossRef]
47. Bharati, S.; Mondal, M.R.H.; Podder, P.; Prasath, V.S. Federated learning: Applications, challenges and future directions. *Int. J. Hybrid Intell. Syst.* **2022**, *18*, 19–35. [CrossRef]
48. Shao, J.; Lin, T.; Cao, X.; Luo, B. Federated Unlearning: A Perspective of Stability and Fairness. *arXiv* **2024**, arXiv:2402.01276.
49. Menditto, A.; Patriarca, M.; Magnusson, B. Understanding the meaning of accuracy, trueness and precision. *Accredit. Qual. Assur.* **2007**, *12*, 45–47. [CrossRef]
50. Jansen, S.C.; Martin, B. The Streisand Effect and Censorship Backfire. *Int. J. Commun.* **2015**, *9*, 16.
51. Serrà, J.; Surís, D.; Miron, M.; Karatzoglou, A. Overcoming catastrophic forgetting with hard attention to the task. *arXiv* **2018**, arXiv:1801.01423.
52. Qu, Y.; Yuan, X.; Ding, M.; Ni, W.; Rakotoarivelo, T. Learn to unlearn: Insights into machine unlearning. *IEEE Computer* **2024**, *57*, 79–90.
53. Wang, R. Redefining Machine Unlearning: A Conformal Prediction-Motivated Approach. *arXiv* **2025**, arXiv:2501.19403.
54. Li, J.; Wei, Q.; Zhang, C.; Qi, G.; Du, M.; Chen, Y. Single Image Unlearning: Efficient Machine Unlearning in Multimodal Large Language Models. *arXiv* **2024**, arXiv:2405.12523.
55. Shrivastava, A.; Ahirwal, M.K. Refining of Learning in Human Decision Making Models: A Step Towards Machine Unlearning. In Proceedings of the 2024 International Conference on Integrated Circuits, Communication, and Computing Systems (ICIC3S), Una, India, 8–9 June 2024.
56. Baumhauer, T.; Schöttle, P.; Zeppelzauer, M. Machine Unlearning: Linear Filtration for Logit-based Classifiers. *arXiv* **2020**, arXiv:2002.02730.
57. Chen, R.; Yang, J.; Xiong, H.; Bai, J.; Hu, T.; Hao, J.; Feng, Y.; Zhou, J.T.; Wu, J.; Liu, Z. Fast Model Debias with Machine Unlearning. *arXiv* **2023**, arXiv:2310.12560.
58. Jia, J.; Liu, J.; Ram, P.; Yao, Y.; Liu, G.; Liu, Y.; Sharma, P.; Liu, S. Model Sparsity Can Simplify Machine Unlearning. *arXiv* **2024**, arXiv:2304.04934.

59. Loog, M.; Viering, T. A Survey of Learning Curves with Bad Behavior: Or How More Data Need Not Lead to Better Performance. *arXiv* **2022**, arXiv:2211.14061.
60. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. *arXiv* **2015**, arXiv:1512.03385.
61. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2015**, arXiv:1409.1556.
62. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. *arXiv* **2019**, arXiv:1801.04381.
63. Huang, G.; Liu, Z.; van der Maaten, L.; Weinberger, K.Q. Densely Connected Convolutional Networks. *arXiv* **2018**, arXiv:1608.06993
64. Zhou, J.; Li, H.; Liao, X.; Zhang, B.; He, W. A Unified Method to Revoke the Private Data of Patients in Intelligent Healthcare with Audit to Forget. *Nat. Commun.* **2023**, *14*, 6255. [PubMed]
65. Lindstrom, C. Evaluation Metrics for Machine Unlearning. *Preprints* **2024**. [CrossRef]
66. Gündoğdu, E.; Unal, A.; Unal, G. A Study Regarding Machine Unlearning on Facial Attribute Data. In Proceedings of the 2024 IEEE 18th International Conference on Automatic Face and Gesture Recognition (FG), Istanbul, Turkiye, 27–31 May 2024.
67. Zhang, H.; Nakamura, T.; Isohara, T.; Sakurai, K. A Review on Machine Unlearning. *Sn Comput. Sci.* **2023**, *4*, 337.
68. Solanki, R.; Creager, E. Promoting User Data Autonomy During the Dissolution of a Monopolistic Firm. *arXiv* **2024**, arXiv:2411.13546.
69. Li, G. Theoretically-Grounded Efficient Deep Learning System Design. Doctoral Dissertation, University of Texas Institutional Repository, Lubbock, TX, USA, 2024.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.