Machine Unlearning: Solutions and Challenges

Jie Xu, Zihan Wu, Cong Wang *Fellow, IEEE,* and Xiaohua Jia *Fellow, IEEE*

***Abstract*—Machine learning models may inadvertently memorize sensitive, unauthorized, or malicious data, posing risks of privacy violations, security breaches, and performance deterioration. To address these issues, machine unlearning has emerged as a critical technique to selectively remove specific training data points' influence on trained models. This paper provides a comprehensive taxonomy and analysis of machine unlearning research. We categorize existing research into exact unlearning that algorithmically removes data influence entirely and approximate unlearning that efficiently minimizes influence through limited parameter updates. By reviewing the state-of-the-art solutions, we critically discuss their advantages and limitations. Furthermore, we propose future directions to advance machine unlearning and establish it as an essential capability for trustworthy and adaptive machine learning. This paper provides researchers with a roadmap of open problems, encouraging impactful contributions to address real-world needs for selective data removal.**

***Index Terms*—Machine Unlearning; Machine Learning Security and Privacy; the Right to be Forgotten**

I. Introduction

THE rapid expansion of Machine Learning (ML) has led to remarkable advancements in tasks such as image recognition [1], natural language processing [2], and recommendation systems [3], significantly impacting various aspects of people's lives [4]. However, the wide adoption of machine learning has raised significant concerns about potential privacy risks, security vulnerabilities, and accuracy deterioration in dynamic settings [5]–[7].

To address these issues, machine unlearning has emerged as a promising technique. Machine unlearning refers to the process of selectively removing the influence of specific training data points on an already trained machine learning model, making the updated model behave the same as a model that was never trained on that data [8]. It provides a subtractive capability to adapt models by removing unauthorized, malicious, or outdated data points without full retraining.

Machine unlearning facilitates several critical applications. First, machine unlearning enforces privacy regulations and protects user privacy. Laws such as the European Union (EU) *General Data Protection Regulation* (GDPR) [9] and *California Consumer Privacy Act* (CCPA) [10] have introduced the Right to Be Forgotten, allowing users to request removing their personal data from companies' trained models [11], [12]. By enabling selective data removal, machine unlearning provides a practical way to enforce these legal rights regarding personal data removal.

Second, machine unlearning enhances model security and robustness against adversarial attacks. Machine learning models are vulnerable to adversarial attacks such as data poisoning attacks, where adversaries inject crafted malicious data into the training set to manipulate the model's behavior [13], [14]. By removing harmful manipulated data points, machine unlearning helps defend models against such vulnerabilities.

Third, machine unlearning improves the adaptability of models over time in dynamic environments. Models trained on static historical data can become outdated as the data distributions shift over time [15]. For example, customer preferences may change in the recommendation system. By selectively removing outdated or unrepresentative data, machine unlearning enables the model to maintain performance even as the environment evolves [16].

Based on the degree of influence removal achieved, machine unlearning methods can be categorized into *Exact Unlearning* and *Approximate Unlearning* [17]. Exact unlearning aims to completely remove the influence of specific data points from the model through algorithmic-level retraining [17], [18]. The advantage is the model will behave as if the unlearned data had never been seen. While providing strong guarantees of removal, exact unlearning usually demands extensive computational resources and is primarily suitable for simpler models. On the other hand, approximate unlearning focuses on efficiently minimizing the influence of target data points through limited parameter-level updates to the model [17], [18]. While not removing influence entirely, approximate unlearning significantly reduces computational and time costs. It enables practical unlearning applications even for large-scale and complex machine learning models, where exact retraining is infeasible.

In this paper, we provide a comprehensive overview of machine unlearning, covering pioneering and state-of-the-art techniques for both exact and approximate unlearning. We critically analyze existing research, highlight advantages and limitations, identify research gaps, and suggest future directions. Our goal is to provide a useful roadmap to make further impactful contributions that address real-world needs for adaptive and trustworthy machine learning systems.

The main contributions of this paper are:

- We provide a comprehensive taxonomy and structured overview of machine unlearning research, categorizing techniques into exact and approximate methods. The taxonomy covers a broad range of existing works, establishing a structured understanding of this emerging field for researchers.
- We give an in-depth critical analysis of existing machine unlearning techniques, highlighting their strengths, limitations, and challenges. This analysis provides valuable insights into theoretical and practical obstacles, guiding future research toward impactful open problems.
- We identify the critical issues on machine unlearning and suggest promising future research directions to advance machine unlearning. These suggestions expand the appli-

cability of machine unlearning and address its limitations effectively.

The remainder of this paper is organized as follows. Section II provides background and preliminaries. Section II-C5 discusses Naive Retraining, while Section III covers exact unlearning, and Section IV explores approximate unlearning. Section V offers a comprehensive discussion on critical issues of machine unlearning. Section VI outlines potential future research directions. Finally, Section VII concludes by summarizing key findings.

II. Background and Preliminaries

A. Machine Learning

Machine learning develops algorithms that allow computers to automatically learn and improve from experience based on data [4]. Generally, machine learning approaches can be categorized into three main types: supervised learning, unsupervised learning, and reinforcement learning [19], [20].

In supervised learning, we are given a training dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ consisting of n input-output pairs. Here, $\mathbf{x}_i \in \mathcal{X}$ denotes the input feature vector from the input space $\mathcal{X} \subset \mathbb{R}^d$, where d is the dimension of the input features. $y_i \in \mathcal{Y}$ is the corresponding output variable from the output space $\mathcal{Y}$. The goal is to learn a model function $\mathcal{M}$ parameterized by $\mathbf{w}$ that maps input $\mathbf{x}$ to outputs y. This is achieved by minimizing a loss function $F(\mathcal{D}; \mathbf{w}) = \sum_{z \in \mathcal{D}} f(z; \mathbf{w})$ that quantifies the discrepancy between the predicted outputs $\hat{y}_i = \mathcal{M}(\mathbf{x}_i; \mathbf{w})$ and the true outputs y_i over the training data. The optimal parameters $\mathbf{w}^*$ are obtained by:

$$\mathbf{w}^* = \text{argmin}_{\mathbf{w} \in \mathcal{H}} F(\mathcal{D}; \mathbf{w}) \tag{1}$$

where $\mathcal{H}$ denotes the hypothesis space containing candidate models.

1) Ensemble Learning: Ensemble learning combines multiple individual models, denoted as weak learners, together to improve prediction and decision-making [21]. By exploiting complementary knowledge and reducing bias and variance, ensemble methods can achieve greater accuracy and robustness compared to single models [22]. The diversity and competence of the component models, size and quality of training data, and ensemble techniques used are key factors determining effectiveness [23]. These principles of ensemble learning have also been adapted and applied in various designs for exact unlearning.

2) Explainable AI: Explainable Artificial Intelligence (AI) techniques aim to increase the transparency and explainability of complex models, making the relationship between training data and model predictions clear [24], [25]. This facilitates us to understand how removing specific training data affects model predictions [26].

Influence function is a tool in explainable machine learning, which quantifies the influence of individual training points on a model's predictions [27]. The influence of a point $z' \in \mathcal{D}$ on model parameters $\mathbf{w}$ can be assessed by slightly increasing its weight during training:

$$\hat{\mathbf{w}}_{\epsilon;z'} \overset{\text{def}}{=} \arg\min_{\mathbf{w} \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} f\left(z_i; \mathbf{w}\right) + \epsilon f(z'; \mathbf{w}). \tag{2}$$

The influence function $\mathcal{I}$ measures the parameter change with respect to changes in the weight of z' and is shown as Eq.(3).

$$\mathcal{I}(z'; f, \hat{\mathbf{w}}, \mathcal{D}) \overset{\text{def}}{=} \left.\frac{\mathrm{d}\hat{\mathbf{w}}_{\epsilon;z'}}{\mathrm{d}\epsilon}\right|_{\epsilon=0} = -H_{\hat{\mathbf{w}}}^{-1} \nabla_{\mathbf{w}} f(z'; \hat{\mathbf{w}}), \tag{3}$$

where $H_{\hat{\mathbf{w}}}$ is the Hessian, $f(\cdot; \hat{\mathbf{w}})$ is the loss function, and $\nabla_{\mathbf{w}} f(\cdot; \hat{\mathbf{w}})$ is the gradient of the loss function. By setting $\epsilon = -1/n$, influence functions can approximate the effect of removing z' without retraining [26].

B. Machine Learning Attacks

1) Data Poisoning Attacks: Data poisoning attacks inject manipulated data into the training dataset with the goal of causing the resulting models to make skewed predictions [13], [14]. This exploits the reliance of statistical models on input data for learning by tampering with the training data. Common techniques include adding noisy samples, flipping output labels, and altering feature values [28]. For example, the attacker can insert poisoned points near the decision boundary to increase test errors or modify features of particular inputs to induce misclassifications. The model trained on the poisoned data will then make predictions based on the contaminated dataset, which can lead to distorted or incorrect results that undermine the integrity of the model [13].

2) Membership Inference Attacks: Membership Inference Attacks (MIAs) aim to determine whether a given data record was part of the training set of a machine learning model [29], [30]. They exploit overfitting, as overfitted models tend to have higher confidence in inputs from their training set [31]. The attacker trains shadow models on their labeled data, then compares the target model's confidence score on the input to the shadow models' scores. Higher confidence in the target model indicates the input likely belonged to its training set. Thus, MIAs can reveal sensitive training data, posing a threat to privacy.

C. Machine Unlearning

1) Problem Definition: Machine unlearning refers to the process of removing the influence of specific training data points on an already trained machine learning model [32]. Formally, given a model with parameters $\mathbf{w}^*$ trained on dataset $\mathcal{D}$ using learning algorithm $\mathcal{A}$, and a subset $\mathcal{D}_f \subseteq \mathcal{D}$ to be removed, the machine unlearning algorithm $\mathcal{U}(\mathcal{A}(\mathcal{D}), \mathcal{D}, \mathcal{D}_f)$ aims to obtain new parameters $\mathbf{w}^-$ by removing the effects of $\mathcal{D}_f$ while preserving performance on $\mathcal{D} \setminus \mathcal{D}_f$.

2) Application Scenarios of Machine Unlearning: Machine unlearning enables selectively removing specific data points from trained ML models. This subtractive capability supports key applications:

- Privacy Protection: Machine unlearning helps enforce privacy rights and enhances privacy protection [5], [33]. It allows removing users' personal data from trained models to comply with regulations such as GDPR and CCPA, which grant users the right to remove their data [33], [34]. By removing training data points, machine unlearning also reduces the attack surface for membership

inference attacks that aim to determine if certain data was used in training [35].
- Improving Security: Machine learning models can be vulnerable to data poisoning attacks, where adversaries inject malicious data into the training set to manipulate the model's behavior. Machine unlearning improves security by removing poisoned data points, making the system more robust against such attacks [28], [35], [36].
- Enabling Adaptability: Models trained on static datasets may not adapt well as the underlying data distribution changes over time, resulting in outdated or inaccurate models [15], [37]. Machine unlearning facilitates adaptability by removing outdated or unrepresentative data, keeping the model relevant even as the environment evolves [16].

3) Challenges in Machine Unlearning: Machine unlearning faces challenges from inherent properties of ML models as well as practical implementation issues:

- Data Dependencies: ML models do not simply analyze data points in isolation. Instead, they synergistically extract complex statistical patterns and dependencies between data points [38]. Removing an individual point can disrupt the learned patterns and dependencies, potentially leading to a significant decrease in performance [16], [32], [39].
- Model Complexity: Large machine learning models such as deep neural networks can have millions of parameters. Their intricate architectures and nonlinear interactions between components make it hard to interpret the model and locate the specific parameters most relevant to a given data point [16], [40]. The lack of transparency into how data influences predictions poses challenges for removing dependencies.
- Computational Cost: Most machine unlearning techniques require iterative optimization methods such as gradient descent to adjust parameters after removing data. This incurs a significant computational cost, which grows rapidly as the model and dataset size increase [16], [41]. The computational demands may exceed the available resources when dealing with large-scale datasets and complex models.
- Privacy Leaks: The unlearning process itself can leak information in multiple ways [42]. Statistics such as the time taken to remove a point can reveal information about it [32], [42]. Changes in accuracy and outputs can also allow adversaries to infer removed data characteristics [32], [43].
- Dynamic Environments: Tracing each data point's influence becomes increasingly difficult as the dataset changes dynamically [11], [37]. Unlearning can also introduce delays that impede prompt model updates needed for low-latency predictions.

4) Metrics for Machine Unlearning Algorithms: Several metrics have been proposed for evaluating the performance of machine unlearning algorithms.

*a) **Completeness**:* This metric evaluates how thoroughly the unlearning algorithm makes the model remove the target data. It compares the model's predictions or parameters before and after unlearning to quantify the extent of removing. Various distance or divergence measures can be used to quantify the difference between the two models [44]. Representative measures include L_2 distance and KL divergence.

L_2 distance measures parameter difference between models $\mathcal{M}_{\mathbf{w_1}}$ with parameters $\mathbf{w}_1$ and $\mathcal{M}_{\mathbf{w_2}}$ with parameters $\mathbf{w}_2$:

$$L_2(\mathcal{M}_{\mathbf{w_1}}, \mathcal{M}_{\mathbf{w_2}}) = \sqrt{\sum_{i=1}^{n} (\mathbf{w}_1 - \mathbf{w}_2)^2}. \tag{4}$$

Smaller L_2 distances indicate higher similarity, while larger distances indicate greater dissimilarity [45].

KL divergence measures difference between prediction distributions before unlearning P and after unlearning Q [46]:

$$\mathrm{KL}(P||Q) = \sum P(x) \log \left(\frac{P(x)}{Q(x)} \right) \tag{5}$$

A smaller KL divergence indicates that the unlearning process made the model remove the specific data more successfully.

*b) **Time Efficiency**:* It can be quantified by comparing the ratio of time required for naive retraining to the time taken for unlearning [47]:

$$E(\mathcal{U}) = \frac{\text{Time for } \mathcal{A}(\mathcal{D} \setminus z')}{\text{Time for } \mathcal{U}}, \tag{6}$$

where $\mathcal{A}(\mathcal{D} \setminus z')$ represents the model after naive retraining, and $\mathcal{U}$ represents the model after unlearning. Higher efficiency scores indicate faster unlearning. This measure becomes particularly relevant in real-time and dynamic scenarios where rapid unlearning is required to enable agile and responsive systems [16].

*c) **Privacy**:* Certified removal [48] is an important privacy metric for approximate unlearning solutions inspired by differential privacy [49]. It offers a theoretical assurance that a model, after specific data removal, is indistinguishable from a model that was never trained on that data. This property implies that an adversary cannot extract information about the removed training data from the model, rendering membership inference attacks on the removed data unsuccessful. This property can be represented in two ways: ϵ-certified removal and its more relaxed version, (ϵ, δ)-certified removal.

Definition 1 (ϵ-certified removal)**.** *Given $\epsilon > 0$, a learning algorithm $\mathcal{A}(\cdot)$, a dataset $\mathcal{D}$, a training data point to remove $z' = (x', y')$, a subset of the hypothesis set $\mathcal{H}$, and a removal mechanism $\mathcal{U}$, the ϵ-certified removal is defined as:*

$$e^{-\epsilon} \leq \frac{P(\mathcal{U}(\mathcal{A}(\mathcal{D}), \mathcal{D}, z') \in \mathcal{H})}{P(\mathcal{A}(\mathcal{D} \setminus z') \in \mathcal{H})} \leq e^{\epsilon}. \tag{7}$$

Definition 2 ((ϵ, δ)-certified removal)**.** *Given $\epsilon > 0$ and $\delta > 0$, the (ϵ, δ)-certified removal is defined as:*

$$\begin{aligned} &P(\mathcal{U}(\mathcal{A}(\mathcal{D}), \mathcal{D}, z') \in \mathcal{H}) \leq e^{\epsilon} P(\mathcal{A}(\mathcal{D} \setminus z') \in \mathcal{H}) + \delta, \\ &P(\mathcal{A}(\mathcal{D} \setminus z') \in \mathcal{H}) \leq e^{\epsilon} P(\mathcal{U}(\mathcal{A}(\mathcal{D}), \mathcal{D}, z') \in \mathcal{H}) + \delta. \end{aligned} \tag{8}$$

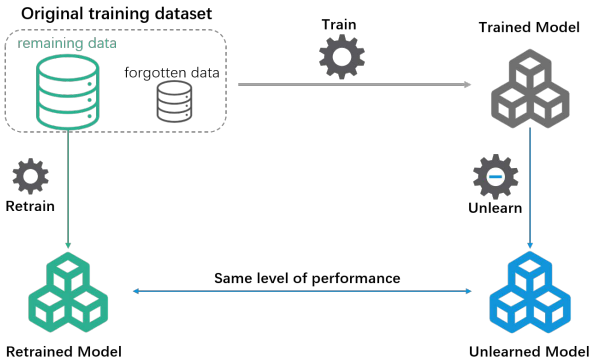

Fig. 1: Illustration of Naive Retraining and Machine Unlearning

5) Naive Retraining: Naive retraining, also known as retraining from scratch, is to remove the data point from the training dataset and retrain the model again. It is often used as a baseline to evaluate unlearning techniques.

Definition 3 (Naive Retraining)**.** *Given a learning algorithm $\mathcal{A}(\cdot)$, dataset $\mathcal{D}$, and a training data point $z' = (x', y')$ to be removed, naive retraining involves retraining on the modified dataset $\mathcal{D} \setminus z'$. Mathematically, it can be represented as:*

$$\mathcal{A}(\mathcal{D} \setminus z'). \tag{9}$$

As shown in Figure 1, naive retraining first discards the existing model. Then, a new model is trained from scratch with the remaining training data after removing the data. This involves initializing a model with random parameters and training it on the remaining data using an appropriate learning algorithm to minimize a loss function. By excluding the target data, the retraining process removes information embedded in the model.

Naive retraining presents several drawbacks: it is computationally intensive due to complete parameter re-optimization, time-consuming for complex models and large datasets, and relies on the original training data, which limits its feasibility when access is restricted.

III. Exact Unlearning

Exact unlearning takes a more efficient strategy than naive retraining. This approach specifically isolates the influence of particular data points on the model, necessitating only the retraining of the affected components instead of the entire model. In this section, we present an overview of exact unlearning through the SISA framework, followed by methods based on the SISA framework and other variations of exact unlearning.

A. Overview of Exact Unlearning

The Sharding, Isolation, Slicing, and Aggregation (SISA) [41] framework is a general approach for exact unlearning. By sharding, isolating, slicing, and aggregating training data, SISA enables targeted data removal without full retraining. It provides an efficient solution considering the tradeoffs in machine unlearning systems.

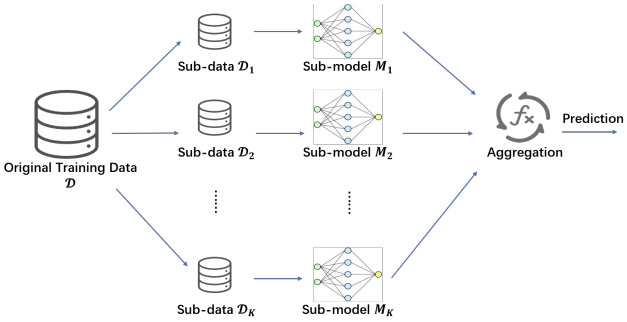

Fig. 2: SISA Framework

The key idea of SISA is to divide the training data into multiple disjoint shards, with each shard for training an independent sub-model. The influence of a data point is isolated within the sub-model trained on its shard. When removing a point, only affected sub-models need to be retrained.

As shown in Figure 2, the implementation of SISA includes four key steps.

(1) Sharding: The training data is divided into multiple disjoint subsets, called 'shards'. Each shard is used to train a separate sub-model.
(2) Isolation: The sub-models are trained independently of each other, ensuring that the influence of a data point is isolated to the model trained on the shard containing that data point.
(3) Slicing: Within each shard, the data is further divided into 'slices'. Models are incrementally trained on these slices. The parameters are stored before including each new slice to track the influence of unlearned data points at a more fine-grained level.
(4) Aggregation: The sub-models trained on each shard are aggregated to form the final model. Aggregation strategies, such as majority voting, allow SISA to maintain good performance.

When unlearning a specific data point, only the sub-models associated with shards containing that data need retraining. The retraining can start from the last parameter saved that does not include the data point to be unlearned.

SISA offers several advantages over naive retraining. First, it reduces the computational cost and time required for unlearning by training models on smaller shards, retraining only the affected models, and incrementally updating models using slices. Second, it maintains prediction accuracy by aggregating the knowledge of the sub-models. Third, SISA provides a flexible and scalable solution, allowing the system to handle evolving unlearning requests without compromising the model's overall performance.

However, SISA does have limitations. First, the effectiveness of SISA depends on the specific characteristics of the learning algorithm of sub-models and the data. For example, it may not work well for models that learn complex interactions between data points or for data that is not easily divisible into independent shards. Second, SISA requires additional storage resources for keeping separate sub-models and tracking each data point's influence within each slice. Third, the model's

generalization ability could be degraded due to isolated training and the tradeoffs involved in aggregation strategies.

In the next subsection, we discuss some notable improvements and adaptations of SISA for different types of models, including random forest, graph-based models, and k-means.

B. Exact Unlearning based on SISA Structure

1) Exact Unlearning for Random Forest: Exact unlearning for random forest can be seen as a specific application of the SISA framework. Each tree in the forest is trained on a different subset of data, acting as a shard in the SISA framework. The predictions of the individual trees are aggregated to obtain the final prediction of the random forest. The influence of a data point is isolated within the trees trained on the subset containing that data point. When unlearning a data point, only the trees trained on the relevant subset require retraining.

DaRE [50] proposes a variant of random forest called Data Removal-Enabled (DaRE) forest. DaRE forest uses a two-level approach with random and greedy nodes in the tree structure. Random nodes, located in upper levels, choose split attributes and thresholds uniformly at random, requiring minimal updates as they are less dependent on data. Greedy nodes, found in lower levels, optimize splits based on criteria such as the Gini index or mutual information. DaRE trees cache statistics at each node and training data at each leaf, allowing for efficient updates of only necessary subtrees when data removal requests are received. This caching and use of randomness improve the efficiency of unlearning.

HedgeCut [39] focuses on unlearning requests with low latency in extremely randomized trees. It introduces the concept of *split robustness* to identify split decisions that may change with removed data. HedgeCut maintains *subtree variants* for such cases, and when unlearning a data point, it replaces the corresponding split with the corresponding subtree variants. This operation is quick and straightforward, ensuring a short delay in the unlearning process.

2) Exact Unlearning for Graph-based Model: The interconnected structure of graph data makes it challenging for graph-based model unlearning, as influence from any single data point spreads across the entire graph. This necessitates the development of specialized graph-based unlearning methods. Exact unlearning for graph-based models aims to efficiently and accurately remove the influence of individual data points on model predictions while accounting for the unique characteristics of graph-structured data.

Two pioneering graph-based unlearning methods, GraphEraser [51] and RecEraser [52], extend the SISA framework to graph data but use different partitioning and aggregation strategies. GraphEraser is designed for Graph Neural Networks (GNNs) unlearning. It consists of three phases: balanced graph partition, shard model training, and shard model aggregation. The graph partition algorithms focus on preserving the graph structural information and balancing the shards resulting from the graph partition. The learning-based aggregation method optimizes the importance score of the shard models to improve the global model utility. When a node needs to be unlearned, GraphEraser removes the node from the corresponding shard and retrains the shard model.

While GraphEraser handles general graph data, it may be less optimal for recommendation systems, where collaborative information across users and items is crucial. RecEraser [52] is specialized for recommendation tasks, where user-item interactions are represented in graphs. It extends the SISA framework by proposing three data partition methods based on users, items, and interactions to divide training data into balanced groups. RecEraser uses an adaptive aggregation method to combine the predictions of the sub-models. This considers both the local collaborative information captured by each sub-model and the global collaborative information captured by all sub-models. Upon receiving a data unlearning request, only the corresponding submodel and aggregation need retraining in RecEraser. Consequently, RecEraser can make accurate recommendations after unlearning user-item interactions compared to the static weight of sub-models in GraphEraser.

3) Exact Unlearning for k-means: DC-k-means [53] adopts the SISA framework but uses a tree-like hierarchical aggregation method. The training data is randomly divided into multiple subsets, each represented by a leaf node in a perfect w-ary tree of height h. A k-means model is trained on each subset, with each leaf node corresponding to a k-means model. The final model is an aggregation of all the k-means models at the leaf nodes of the tree, achieved by recursively merging the results from the leaf nodes to the root. To unlearn a data point, the relevant leaf node is located, and the corresponding k-means model is updated to exclude that data point. The updated model then replaces the old model at the leaf node, and the changes propagate up the tree to update the final aggregated model.

4) Exact Unlearning for Federated Learning (FL): KNOT [54] adopts the SISA framework for client-level asynchronous federated unlearning during training. A clustered aggregation mechanism divides clients into multiple clusters. The server only performs model aggregation within each cluster, while different clusters train asynchronously. When a client requests to remove its data, only clients within the same cluster need to be retrained, while other clusters are unaffected and can continue training normally. To obtain an optimal client-cluster assignment, KNOT formulates it as a lexicographic minimization problem. The goal is to minimize the match rating between each client and assigned cluster, considering both training speed and model similarity. This integer optimization problem can be efficiently solved as a Linear Program(LP) using an off-the-shelf LP solver.

5) Improvements of SISA: ARCANE [18] is designed to overcome the limitations of SISA, aiming to accelerate the exact unlearning process and ensure retrained model accuracy. Unlike SISA's random and balanced data partition, ARCANE divides the dataset into class-based subsets, training sub-models independently using a one-class classifier. This approach reduces accuracy loss by confining unlearning influence to a single class. ARCANE also introduces data preprocessing methods to reduce retraining costs. These methods include representative data selection, model training state saving, and data sorting by erasure probability. Representative data selection removes redundancies and focuses on selecting the

TABLE I: Exact Unlearning Methods based on SISA

Paper	Goal	Design Ideas	Strengths	Weaknesses
SISA [41]	Efficient unlearning for general models	• Sharded, isolated, sliced, aggregated training	• Applicable to any model • Improves unlearning efficiency • Scalable	• Breaking data dependencies • Additional storage cost for model parameters • Accuracy degradation
DaRE [50]	Efficient unlearning for random forest	• Use cached statistics to only retrain affected subtrees • Consider subsets of thresholds per attribute	• Achieves efficiency by retraining affected subtrees • Analyzes the effect of layers and thresholds	• Additional storage cost for caching statistics • Worst case performance no better than retraining
HedgeCut [39]	Low-latency unlearning for ensemble of random forest	• Use split robustness to identify splitting decisions • Prepare subtree variants for non-robust splits	• Low-latency unlearning • Predictive accuracy similar to random forest	• Rely on the accuracy of predicted unlearning requests • Storage cost for subtree variants
DC-k-means [53]	Efficient unlearning for k-means	• Divide-and-Conquer approach	• Theoretical guarantees on removal efficiency • Negligible quality loss	• State storage cost • High removing time complexity in high dimensions
GraphEraser [51]	Efficient unlearning for for GNNs	• Community detection for balanced sharding • Shard model importance scores	• Maintains graph structure • Balanced partitioning ensures unlearning efficiency	• Additional cost of maintaining massive shards. • Not satisfy the adaptive setting
RecEraser [52]	Efficient unlearning for recommender systems	• Different data partition strategies • Attention-based adaptive aggregation method	• Efficient unlearning with balanced data partition • Good global model utility with adaptive aggregation	• Specific to recommendation task • Open problem in the batch setting
KNOT [54]	Federated unlearning during training	• Cluster clients and perform aggregation within clusters • Clusters train asynchronously	• Limits retraining to a cluster • Asynchrony allows unaffected clusters to continue training	• Performance relies on cluster assignments • Asynchrony can negatively impact model accuracy
ARCANE [18]	Efficiently and accurately unlearning	• Multiple one-class classification tasks • Data preprocessing	• Maintains accuracy even with large unlearning requests • Data preprocessing accelerates unlearning	• Only applies to supervised models • Rely on preprocessing for efficiency

most informative subset of the training set. Training state saving allows for the reuse of previous calculation results, further improving efficiency. Additionally, sorting the data by erasure probability enhances the speed of handling unlearning requests.

C. Non-SISA Exact Unlearning

Cao *et al.* [16] were inspired by statistical query learning and proposed an intermediary layer called 'summations' to decouple machine learning algorithms from the training data. Instead of directly querying the data, learning algorithms rely on these summations. This allows the removal of specific data points by updating the summations and computing the updated model. The unlearning process involves two steps. First, the feature set is updated by excluding the data point to be removed and re-scoring the features. The updated feature set is generated by selecting the top-scoring features. This process is more efficient than retraining as it does not require examining each data point for each feature. Second, the model is updated by removing the corresponding data if a feature is removed from the feature set or computing the data if a feature is added. Simultaneously, summations dependent on the removed data point are updated, and the model is adjusted accordingly.

Liu *et al.* [55] propose a rapid retraining approach for FL. When a client requests data removal, all clients perform local data removal, followed by a retraining process on the remaining dataset. This process utilizes a first-order Taylor approximation technique based on the Quasi-Newton method and a low-cost Hessian matrix approximation method. The proposed approach effectively reduces computational and communication costs while maintaining model performance, providing an efficient realization of the right to be forgotten in FL.

D. Comparisons and Discussions

The comparison of different exact unlearning methods is shown in Table I. We evaluate their strengths and weaknesses in terms of storage cost, assumptions, model utility, computational cost, scalability, and practicality. Our goal is to provide a clear assessment to help guide future research and select suitable unlearning methods for different applications.

(1) Additional Storage Cost: Exact unlearning methods rely on substantial additional storage required for caching model parameters, statistics, or intermediate results. For instance, SISA requires the storage of model parameters for each shard and slice, while HedgeCut requires the storage of subtrees variants. This hinders scalability to large models or frequent unlearning requests. Developing low-storage approaches would be beneficial.

(2) Strong Assumptions: Some methods make strong assumptions about the learning algorithm or data characteristics. SISA may struggle with highly dependent data, while statistical query learning requires algorithms to be expressed in a summation form. Methods tailored to specific models, such as DaRE and HedgeCut, GraphEraser, and RecEraser, have limited applicability. More flexible and generalizable techniques are needed.

(3) Model Utility: Most methods claim they maintain accuracy after unlearning but lack thorough analysis across diverse settings. Rigorous evaluations of different models, datasets, and removal volumes are imperative to provide concrete utility guarantees.
(4) Computational Cost: Exact unlearning methods add computational costs during initial training because of multiple sub-models training and aggregation. It may not be feasible when computational resources are limited.
(5) Managing Evolving Data: Existing methods focus on removing data in fixed training sets. Handling dynamically changing data with continuous insertion and removal requests remains an open problem needing incremental update mechanisms.

In summary, while existing exact unlearning methods enable efficient and accurate data removal, they have limitations related to storage, assumptions, utility maintenance, and scalability. The most suitable method depends on the specific requirements of the machine learning application, including the data type, model type, and the desired balance between efficiency and accuracy. Further research should aim to develop generally applicable techniques with low cost that provably sustain model accuracy even after repeated unlearning. This will lead to more robust and practical solutions aligned with the evolving privacy needs of machine learning systems.

IV. Approximate Unlearning

Approximate unlearning aims to minimize the influence of unlearned data to an acceptable level while achieving an efficient unlearning process. It has several advantages over exact unlearning techniques, including better computational efficiency, less storage cost, and higher flexibility.

- Computational Efficiency: Exact unlearning methods require retraining at the algorithm level using remaining data, which can be computationally expensive, particularly for large datasets. In contrast, approximate unlearning focuses on minimizing the data's influence to be removed rather than completely removing it. For example, Guo et al. [48] proposed a method that adjusts the model parameters based on the calculated influence of the removed data. This approach is less computationally intensive than the full re-computation required in exact unlearning, leading to increased computational efficiency.
- Storage Overhead: Exact unlearning typically requires storing the training dataset and multiple sub-models, leading to high storage costs. Conversely, approximate unlearning, such as the one proposed by Sekhari *et al.* [56], only requires storing cheap-to-compute data statistics. As a result, the storage burden associated with exact unlearning is significantly alleviated.
- Flexibility: Exact unlearning methods are often tailored to specific learning models or data structures, limiting their applicability. On the other hand, many approximate unlearning are more model-agnostic. They can be applied to a diverse range of learning algorithms without requiring specific model or data structure modifications. This enhanced flexibility allows approximate unlearning to be more widely applicable compared to exact unlearning.

It is important to note that approximate unlearning represents a tradeoff between unlearning completeness and unlearning efficiency [16]. In some cases, a slight decrease in completeness, but a significant speed-up of the unlearning process and savings in computation and storage costs, is an acceptable tradeoff. By carefully considering these tradeoffs, approximate unlearning provides an effective and practical approach for adapting models to new data and tasks.

A. Overview of Approximate Unlearning

Approximate unlearning is a process that aims to minimize the influence of unlearned data to an acceptable level rather than completely removing it. It involves the following key steps:

(1) Computation of Influence: Calculate the influence of the data points that need to be unlearned on the original model. This involves determining how these data points affect the model's predictions.
(2) Adjustment of Model Parameters: Modify the model parameters to reverse the influence of the data being removed. This adjustment typically involves methods such as reweighting or recalculating optimal parameters and modifying the model so that it behaves as if it was trained on the dataset without the unlearned data points.
(3) Addition of Noise: Carefully calibrated noise is added to prevent the removed data from being inferred from the updated model. This step ensures the confidentiality of the training dataset.
(4) Validation of Updated Model: Evaluate the performance of the updated model to ensure its effectiveness. This validation step may involve cross-validation or testing on a hold-out set to assess the model's accuracy and generalization.

By following these steps, approximate unlearning efficiently reduces the influence of specific data points in a trained model. This approach provides a practical alternative to exact unlearning, particularly in scenarios where computational cost, storage cost, and flexibility are crucial factors.

The subsequent subsections provide more details on specific approximate unlearning. Section IV-B discusses methods based on the influence function, while Sections IV-C and IV-D explore methods based on re-optimization and gradient update, respectively. Section IV-E introduces methods specifically designed for graph data, and Section IV-F covers other notable methods in the field. Each section provides a detailed summary of the respective methods, discussing their applications, advantages, and limitations.

B. Approximate Unlearning based on Influence Function of the Removed Data

Influence functions offer a powerful tool for understanding the influence of specific training data points on the predictions of a machine learning model [26]. This understanding facilitates an update to the model that effectively unlearns the influence of certain data points without full retraining. This section explores representative works leveraging influence functions

for approximate unlearning and discusses their applications and limitations.

The core idea shared by these representative works is to use the influence function to estimate data points' influence for removal. Some works [48], [56]–[60] develop unlearning algorithms that approximately compute the influence function of data points on the model parameters, and then update the parameters to remove the influence of unlearned data points. Some [56]–[58] also focus on efficiently approximating the influence functions by considering only relevant parameters or using sampling because the direct computation of influence functions can be computationally expensive.

We start by discussing the pioneering work of Guo *et al.* [48] and its limitations. Then, we will explore how subsequent research has built upon Guo's work to address these limitations, particularly by reducing the computational cost associated with inverting the Hessian matrix. Finally, we will discuss the future directions of this research field.

Guo *et al.* [48] first introduced influence functions for data removal and achieved certified removal of L_2-regularized linear models. Specifically, linear models are usually trained using a differentiable convex loss function as shown in Eq.(10).

$$F(\mathcal{D};\mathbf{w}) = \sum_{z\in\mathcal{D}} f(z;\mathbf{w}) + \frac{\lambda n}{2}\left\|\mathbf{w}\right\|_2^2, \tag{10}$$

where $f(z;\mathbf{w})$ is a convex loss function. To protect the information of the removed data points, Guo *et al.* propose to add random perturbation [61] during the training process to protect the gradient information. Thus, the loss function used for training is as shown in Eq. (11):

$$F_{\mathbf{b}}(\mathcal{D};\mathbf{w}) = \sum_{z\in\mathcal{D}} f(z;\mathbf{w}) + \frac{\lambda n}{2}\left\|\mathbf{w}\right\|_2^2 + \mathbf{b}^\top\mathbf{w}, \tag{11}$$

where $\mathbf{b}$ is a random vector. The parameters of the model is $\mathbf{w}^*$, where

$$\mathbf{w}^* = A(\mathcal{D}) = \mathrm{argmin}_{\mathbf{w}}\, F_{\mathbf{b}}(\mathcal{D};\mathbf{w}). \tag{12}$$

Suppose the data point $z' = (x', y')$ is to be removed from the training set. The process of Newton update removal mechanism to remove z' is as follows:

(1) Calculate the influence of the removed data point on the model parameters. The loss gradient at z' is

$$\Delta = \lambda\mathbf{w}^* + \nabla f(z';\mathbf{w}). \tag{13}$$

Thus, the influence of z' on the original model is $-H_{\mathbf{w}^*}^{-1}\Delta$ [26], where $H_{\mathbf{w}^*}$ is the Hessian of the loss function

$$H_{\mathbf{w}^*} = \nabla^2 F\left(\mathcal{D}_r;\mathbf{w}^*\right), \mathcal{D}_r = \mathcal{D}\setminus z'. \tag{14}$$

This one-step Newton update is applied to the gradient influence of the removed point z'.

(2) Adjust the model parameters $\mathbf{w}^*$ to removes the influence of the z' from the model. The new model parameters $\mathbf{w}^-$ are given by

$$\mathbf{w}^- = \mathbf{w}^* + H_{\mathbf{w}^*}^{-1}\Delta. \tag{15}$$

(3) Perturbation has been added during the training process as shown in Eq.(11).

(4) Validate the new model by checking the removed data point's influence.

Steps (1) and (2) make the new model parameters w^- approximate the original model parameters w^* as closely as possible while ensuring the removal of specific data points from the original dataset. The process also ensures the performance of the modified model and minimizes the leakage of information about the removed data points by adding random perturbations to the model parameters during training and bounding the norm of the gradient residual. Thus, the removal mechanism is a certified-removal mechanism.

While Guo *et al.* [48] have provided valuable insights into approximate unlearning, their proposed removal mechanism has limitations. It requires inverting the Hessian matrix, fails to work for models with non-convex losses, and has a significant gap between the data-dependent bound and the true gradient residual norm. To address these limitations, subsequent research [56]–[58] has made improvements by building on Guo's work.

Building upon Guo's work, Sekhari *et al.* [56] does not require full access to the training dataset during the unlearning process. By using cheap-to-store data statistics $\nabla^2\widehat{F}(\mathcal{D};\mathbf{w}^*)$ as shown in Eq.(16), they enable efficient unlearning without the need for the entire training data reducing computational and storage requirements, in contrast to Eq.(14) and Eq.(15).

$$\begin{aligned}\widehat{H} &= \frac{1}{n-m}\left(n\nabla^2\widehat{F}(\mathcal{D};\mathbf{w}^*) - \sum_{z'\in\mathcal{D}_f}\nabla^2 f(z';\mathbf{w}^*)\right),\\ \mathbf{w}^- &= \mathbf{w}^* + \frac{1}{n-m}(\widehat{H})^{-1}\sum_{z'\in\mathcal{D}_f}\nabla f(z';\mathbf{w}^*).\end{aligned} \tag{16}$$

They also emphasize the importance of test loss and add noise after adjusting model parameters to ensure model performance. This ensures privacy protection without compromising the accuracy and performance of the model.

Suriyakumar *et al.* [57] propose a more computationally efficient algorithm for online data removal from machine learning models trained with empirical risk minimization (ERM). This improvement is achieved by using the *infinitesimal jackknife*, a technique that approximates the influence of excluding a data point from the training dataset on the model parameters. This avoids the need to compute and invert a different Hessian matrix for each removal request, which was required by prior methods [48], [56]. Their approach enables efficient processing of online removal requests while maintaining similar theoretical guarantees on model accuracy and privacy. Moreover, by integrating the infinitesimal jackknife with Newton methods, their algorithm can accommodate ERM-trained models with non-smooth regularizers, broadening applicability.

Mehta *et al.* [58] improve the efficiency of Hessian matrix inversion in deep learning models. They introduce a selection scheme, L-CODEC, which identifies a subset of parameters to update, removing the need to invert a large matrix. This avoids updating all parameters, focusing only on influential ones. Building on this, they propose L-FOCI to construct a minimal

set of influential parameters using L-CODEC incrementally. Once the subset of parameters to update is identified, they apply a blockwise Newton update to the subset. By focusing computations only on influential parameters, their approach makes approximate unlearning feasible for previously infeasible large deep neural networks.

Unlike the approach by Guo *et al.* [48], which only adjusts the linear decision-making layer of a model, PUMA [59] modifies all trainable parameters, offering a more thorough solution to data removal. The main purpose of PUMA is to maintain the model's performance after data removal, rather than just monitoring whether the modified model can produce similar predictions to a model trained on the remaining data, as Guo *et al.*'s method does. To achieve this, PUMA uses the influence function to measure the influence of each data point on the model's performance and then adjusts the weight of the remaining data to compensate for the removal of specific data points. This approach allows for efficient data removal without significant performance degradation.

Tanno *et al.* [60] propose a Bayesian continual learning approach to identify and erase detrimental data points in the training dataset. They use influence functions to measure the influence of each data point on the model's performance, allowing them to identify the most detrimental training examples that have caused observed failure cases. Once these detrimental data points are identified, the model is updated to erase the influence of these points. This is achieved by approximating a "counterfactual" posterior distribution, where the harmful data points are assumed to be absent. The authors propose three methods for updating the model weights, one of which is a variant of the Newton update proposed by Guo *et al.* [48]. This approach effectively removes the influence of the detrimental data points from the model, repairing the model without full retraining.

Warnecke *et al.* [62] point out that unlearning should not be limited to removing entire data points. Instead, it should enable corrections at various granularities within the training data. They propose a method that uses influence functions to remove specific features and labels from a trained model. By reformulating influence estimation as a form of removing, the authors derive an approach that maps changes in training data retrospectively to closed-form updates of model parameters. These updates can be computed efficiently even when large portions of the training data are affected, effectively correcting the problematic features and labels within the model.

Comparisons and Discussions. The influence function was first introduced for efficient data removal by Guo *et al.* [48], providing a one-step Newton update to remove data points based on their influence on model parameters. However, this pioneering work relied on convexity assumptions and suffered from high computational costs due to the need to invert the Hessian matrix. Subsequent research addressed these limitations by developing more efficient approximations of influence functions. The summary and comparison of approximate unlearning based on influence functions are in Table II.

A key challenge in this field is the computational cost associated with inverting the Hessian matrix, a step necessary for estimating the influence of data points and updating model parameters. Several strategies have been proposed to address this issue. Sekhari *et al.* [56] reduced storage and computation costs by avoiding the need for the full training data. Suriyakumar *et al.* [57] proposed using the infinitesimal jackknife technique to efficiently approximate influence in an online setting, also extending unlearning to non-smooth regularizers. Mehta *et al.* [58] introduced a conditional independence-based selection of influential parameters to avoid inverting large matrices, enabling unlearning in deep neural networks. These techniques demonstrate the potential for efficient approximate unlearning in practical scenarios.

Moreover, considerations such as test loss [56], thoroughness of removal [59], and finer-grained corrections [62] indicate that revised algorithms are becoming increasingly suitable for diverse real-world applications. The extension of these methods to non-convex models [57], [62] and deep neural networks [58], [62] further highlights their potential.

However, significant challenges remain to be addressed. The accuracy and efficiency of influence estimation and parameter updating require analysis and could potentially be improved. Furthermore, the optimal selection of parameters for updates and the best techniques for influence estimation remain open questions. Connections to differential privacy and information theory may yield valuable insights into inherent limits.

In summary, approximate unlearning based on influence functions shows promise for efficient data removal. This direction enables important progress on algorithmic data removal and its impacts. With continued research, it is expected to play a key role in addressing privacy regulation challenges in the era of big data.

C. Approximate Unlearning based on Re-optimization after Removing the Data

Approximate unlearning based on re-optimization re-optimizes the model after removing data to minimize the influence of the removed data points while maintaining model performance. The key steps are:

(1) Train a model $\mathcal{M}(\mathbf{x}; \mathbf{w})$ with parameters $\mathbf{w}$ on the full dataset $\mathcal{D}$. The original loss function is F, and the minimum value is obtained at $\mathbf{w}^*$.
(2) Define a loss function $F(\mathcal{D}_r; \mathbf{w})$ that maintain accuracy on remaining data $\mathcal{D}_r$ while removing information about data to be forgotten $\mathcal{D}_f$.
(3) Re-optimize the model from $\mathbf{w}_*$ by finding updated parameters $\mathbf{w}^-$ that minimize $F(\mathcal{D}_r; \mathbf{w})$. The updated model $\mathcal{M}(\mathbf{x}; \mathbf{w}^-)$ retains performance on $\mathcal{D}_r$ while statistically behaving as if trained without $\mathcal{D}_f$.

Research in this area has proposed different techniques to implement the key steps above. They have adopted different techniques for selective removing/forgetting based on application goals.

Golatkar *et al.* [46] propose an optimal quadratic scrubbing algorithm to achieve selective forgetting in deep networks. Selective forgetting is defined as a process that modifies the network weights using a scrubbing function $S(\mathbf{w})$ to make the distribution indistinguishable from weights of a network never trained on the forgotten data. Selective forgetting is

TABLE II: Approximate Unlearning based on Influence Function

Paper	Goal	Design Ideas	Strengths	Weaknesses
[48]	Certified removal	• One-step Newton update/ influence function • Difference privacy	• Efficient training data removal • Strong theoretical certified removal guarantee	• Relies on convexity • High computational cost for inverting the Hessian matrix
[56]	Efficient unlearning with generalization guarantees	• Use influence functions to identify important data points • Store cheap data statistics	• Considers test loss instead of just training loss • Reduce computational and storage costs • Give the deletion capacity	• Relies on convexity • Relies on storage of data statistics • Does not handle finite / discrete hypothesis classes
[57]	Efficient unlearning for ERM models	• Infinitesimal jackknife • Newton update	• Computationally efficient online unlearning • Accommodating non-smooth regularizers	• Specific to ERM models • Inefficient for batch removal
[58]	Efficient unlearning for DNN	• Conditional independence-based parameter selection	• Avoids full Hessian inverse • Improves unlearning efficiency in DNN	• Relies on weighted sampling based on Lipschitz constants of filters/layers • Dependence on newly developed optimization tools
[59]	Maintain performance during unlearning	• Performance unchanged model augmentation	• Maintain performance after removal • Computationally efficient	• Limited evaluation on simple datasets • Sensitive to hyperparameters
[60]	Repair model by data removal	• Identify detrimental data via influence function • Remove via posterior approximation	• Model-agnostic framework • Identify causes of failures	• Limited to detrimental data removal • Cause identification can be computationally intensive
[62]	Unlearn features and labels	• Use influence functions as updates for features or labels	• Effective unlearning of features/labels • Efficient closed-form updates	• Efficacy drops as affected features/labels increase • No guarantee for non-convex models

measured by the Kullback-Leibler (KL) divergence. If the KL divergence between the network weight distribution after selective forgetting and the network weight distribution that has never seen the forgotten data is 0, the two distributions are exactly the same, which indicates complete forgetting.

The authors assume that the quadratic loss function $L(\mathcal{D}; \mathbf{w})$ can be decomposed into:

$$L(\mathcal{D}; \mathbf{w}) = L(\mathcal{D}_f; \mathbf{w}) + L(\mathcal{D}_r; \mathbf{w}), \tag{17}$$

where $L(\mathcal{D}_f; \mathbf{w})$ is the loss of the data to be removed and $L(\mathcal{D}_r; \mathbf{w})$ is the loss of remaining data.

The goal of optimization is to minimize *Forgetting Lagrangian* $\mathcal{L}$:

$$\mathcal{L} = \mathbb{E}_{S(\mathbf{w})}[L(\mathcal{D}_r; \mathbf{w})] + \lambda \, \mathrm{KL}[P(S(\mathbf{w})|\mathcal{D})||P(S_0(\mathbf{w})|\mathcal{D}_r)], \tag{18}$$

where $\mathbb{E}_{S(\mathbf{w})}[L(\mathcal{D}_r; \mathbf{w})]$ is the expected loss on the remaining data, λ is a hyperparameter that trades off residual information about the data to be removed and accuracy on the remaining data.

The robust scrubbing function $S_t(\mathbf{w})$ that efficiently removes information about specific training data is:

$$\begin{aligned} S_t(\mathbf{w}) = \mathbf{w} + e^{-Bt}e^{At}d + e^{-Bt}(d - d_r) - d_r \\ + (\lambda\sigma_h^2)^{1/4}B^{-1/4}n, \end{aligned} \tag{19}$$

where $n \sim N(0, I)$, $A = \nabla^2 L(\mathcal{D}; \mathbf{w}), B = \nabla^2 L(\mathcal{D}_r; \mathbf{w}), d = A^{-1}\nabla_{\mathbf{w}} L(\mathcal{D}; \mathbf{w})$ and $d_r = B^{-1}\nabla_{\mathbf{w}} L(\mathcal{D}_r; \mathbf{w})$. λ balances information retention and accuracy. The hyperparameter σ_h reflects the error in approximating the Stochastic Gradient Descent (SGD).

Using Fisher Information Matrix (FIM) [63] to approximate the Hessian matrix, $S_t(\mathbf{w})$ simplifies to:

$$S(\mathbf{w}) = \mathbf{w} + (\lambda\sigma_h^2)^{1/4}\mathcal{F}^{-1/4}, \tag{20}$$

where $\mathcal{F}$ is the FIM computed at $\mathbf{w}$ for $\mathcal{D}_r$.

In their follow-up work [64], Golatkar *et al.* note that weight changes may not affect deep network outputs due to overparameterization. Consequently, attackers could still extract removed data $\mathcal{D}_f$ from the outputs. To address this issue, they focus on *final activations*. These activations represent the model's response to input data and more directly reflect memory and removing processes. They use a Neural Tangent Kernel (NTK) to correlate weights and activations, and introduce an NTK-based scrubbing process to achieve removing by minimizing the difference between the activation of the network on the removing dataset and the target model.

Later, Golatkar *et al.* [65] consider mixed-privacy settings where only some user data needs to be removed, while core data are retained. The key insight is to separate the model into two sets of weights: Non-linear core weights trained conventionally on the core data, ensuring they only contain knowledge from the core data that does not need to be removed. Linear user weights are obtained by minimizing a quadratic loss on all user data. To remove a subset of user data, the optimal user weight update is directly computed by minimizing the loss on the remaining user data. This aligns with the theoretical optimal update for quadratic loss functions and achieves efficient, accurate removal in mixed-privacy settings without reducing core data accuracy.

TABLE III: Approximate Unlearning based on Re-optimization

Paper	Goal	Design Ideas	Strengths	Weaknesses
[46]	Selective forgetting in deep networks	• Optimal quadratic scrubbing on weights • Add noise tailored to the loss landscape	• Formal definition of selective forgetting • Provides upper bound on remaining info	• Rely on the stability of SGD • Computationally expensive, limited scalability
[64]	Selective forgetting from input-output observations	• Analysis based on final activations • NTK-based scrubbing	• Provides tighter black-box bounds with limited queries • Handles over-parameterized models	• Relies on linearization assumptions • Computationally expensive
[65]	Selective forgetting for large deep networks	• Mixed-privacy setting • User weights obtained by minimizing quadratic loss	• Maintaining accuracy on large vision tasks • Provides information bounds	• Relies on the strong convexity assumptions • Forgetting quality depends on data subsets
[66]	Class-level selective forgetting in lifelong learning	• Mnemonic codes • New loss function with four components	• Class-level removing • No need for original data	• Customized for image data • Computational cost of embedding codes

Shibata *et al.* present Learning with Selective Forgetting (LSF) [66] to achieve class-level selective forgetting in lifelong learning. They introduce a loss function F with four components: classification loss F_C to ensure the accuracy of classification, mnemonic loss F_M tying each class to an embedded code, selective forgetting loss F_{SF} to remove classes marked for removal, and regularization loss F_R to prevent catastrophic forgetting.

$$F = \overbrace{F_C + F_M}^{\text{new task}} + \overbrace{F_{SF} + F_R}^{\text{previous tasks}}. \tag{21}$$

The mnemonic codes allow selective forgetting of any class by discarding its code without the original data. The model can selectively forget certain classes by re-optimizing on new loss as illustrated in Eq.(21) without the mnemonic codes of classes to be forgotten.

Comparisons and Discussions. Recent research on selective forgetting for deep neural networks has pursued re-optimization strategies to update models and reduce the influence of data points to be removed. These works have shown promising progress in approximate unlearning through re-optimization techniques, while also revealing key challenges and opportunities for improvement. The summary and comparison of approximate unlearning based on re-optimization are in Table III.

Earlier work [46] introduced core techniques such as weight scrubbing and adding noise, providing a theoretical framework to bound residual information. However, as noted in their late work [64], these weight-centric analyses failed to fully capture the information remaining in activations, leading to cleaner removal guarantees by analyzing outputs.

Subsequent work has aimed to improve computational efficiency and accuracy. Approximations using the Fisher information matrix [46] or NTK [64] help address scalability but may still be expensive and rely on simplifying assumptions. Work by [65] separates weights into fixed core and trainable user components. By optimizing a quadratic loss, efficient removing procedures for user weights could be derived. However, reliance on strong convexity and linear approximations may limit generalizability. Concurrently, research [66] used memory codes to enable class-level removing, but stability and transferability over multiple tasks remain unproven.

Another key challenge is quantifying the closeness between original and re-optimized models, which impacts unlearning effectiveness. The more similar the re-optimized model is to one trained without the data to be removed, the less extractable information remains. However, accurately bounding this similarity is difficult for complex deep networks. Developing rigorous verification methods is an important open problem.

In addition, these approaches make simplifying assumptions about training processes and loss landscapes that may not fully capture the intricate behaviors of large non-convex models optimized with SGD. Relaxing assumptions such as quadratic losses could improve generalization.

Finally, inherent tradeoffs exist between privacy, accuracy, and efficiency in approximate unlearning. Performance impact should be minimized, but some degradation is typically unavoidable. Tuning this balance optimally for different applications remains an open research problem. The isolation of user weights [65] is an interesting architectural adaptation that could enable broader applications for compartmentalizing model knowledge. Extending this approach merits further exploration.

Overall, approximate unlearning based on re-optimization shows promising early progress on efficient data removal for deep learning. This direction enables valuable advancement in algorithmic data removal and its impacts. Advancing theoretical foundations, scaling to large models, and developing robust algorithms tailored to complex training dynamics could further accelerate progress in this critical area.

D. Approximate Unlearning based on Gradient Update

The model parameters need only small adjustments in machine unlearning because the variation of the dataset is often small. Approximate unlearning based on gradient updates makes small adjustments to model parameters to modify the model after removing or adding data points incrementally. This avoids the computational expense of full retraining. Approximate unlearning based on gradient update generally follows a two-step framework to update trained models after minor data changes without full retraining:

TABLE IV: Approximate Unlearning based on Gradient Update

Paper	Goal	Design Ideas	Strengths	Weaknesses
DeltaGrad [45]	Efficiently retrain models after minor data changes	• Cached training parameters and gradients • Burn-in iterations + L-BFGS approximation	• Applicable to general models with GD/SGD • Support both addition and removal	• Gradient storage cost • Relies on strong convexity assumptions
FedRecover [67]	FL model recovery after poisoning attacks	• Server estimates updates • L-BFGS approximation +corrections	• Estimate clients' model updates to reduce communication cost • Scalable to numerous clients	• Storing historical information • Convexity assumptions
Descent-to -Delete [68]	Unlearning with efficiency and privacy guarantees	• Gradient descent perturbations • Data partitioning	• Gaussian noise for indistinguishability • Handles arbitrary updates • Improved accuracy for high-dimensional data	• Convexity assumptions • Accuracy/efficiency tradeoff
BAERASER [69]	Remove backdoor	• Trigger pattern recovery • Gradient ascent unlearning	• Removes backdoors without retraining data • Prevents catastrophic forgetting	• Recovered triggers not identical • Applicable to backdoor only

(1) Initialize the model parameters using the previously trained model.
(2) Perform a small number of gradient update steps on the new data.

Within this framework, the methods make optimizations and customizations to improve efficiency and accuracy.

DeltaGrad [45], a representative of this category, adapts models efficiently to small training set changes by utilizing cached gradient and parameter information during the original training process. The algorithm includes two cases: burn-in iteration and other iterations.

Before Update: The model parameters $\mathbf{w}_0, \mathbf{w}_1, ..., \mathbf{w}_t$ and corresponding gradients $\nabla F(\mathcal{D}; \mathbf{w}_0), \nabla F(\mathcal{D}; \mathbf{w}_1)$, ..., $\nabla F(\mathcal{D}; \mathbf{w}_t)$ of the training process on the full training dataset are cached.

Burn-in Iteration: The algorithm computes gradients exactly in initial burn-in iterations for correction:

$$\begin{aligned} &\nabla F\left(\mathcal{D}; \mathbf{w}_t^I\right) = \nabla F\left(\mathcal{D}; \mathbf{w}_t\right) + \mathbf{H}_t \cdot \left(\mathbf{w}_t^I - \mathbf{w}_t\right), \\ &\mathbf{w}_{t+1}^I = \mathbf{w}_t^I - \frac{\eta_t}{n-r}\left[\sum_{z \notin U} \nabla f\left(z; \mathbf{w}_t^I\right)\right] \end{aligned} \tag{22}$$

$\mathbf{w}_t^I$ denotes the updated model parameter, and $\mathbf{H}_t = \int_0^1 \mathbf{H}\left(\mathbf{w}_t + x\left(\mathbf{w}^I{}_t - \mathbf{w}_t\right)\right) \mathrm{d}x$ is the integrated Hessian matrix at iteration step t.

Other Iteration: The algorithm approximates $\mathbf{H}_t$ using the L-BFGS algorithm [70] and uses this approximation $\mathbf{B}_t$ to compute updated gradients:

$$\begin{aligned} &\nabla F\left(\mathcal{D}; \mathbf{w}_t^I\right) = \nabla F\left(\mathcal{D}; \mathbf{w}_t\right) + \mathbf{B}_t \cdot \left(\mathbf{w}_t^I - \mathbf{w}_t\right), \\ &\mathbf{w}_{t+1}^I = \mathbf{w}_t^I - \frac{\eta_t}{n-r}\left[n \nabla F\left(\mathcal{D}; \mathbf{w}_t^I\right) - \sum_{z' \in U} \nabla f\left(z'; \mathbf{w}_t^I\right)\right] \end{aligned} \tag{23}$$

FedRecover [67] takes a similar approach to recover accurate global models from poisoned models in federated learning while minimizing computation and communication costs on the client side. The key idea is that the server uses the historical information collected during the training of the poisoned global model to estimate the client's model update during recovery. FedRecover utilizes an algorithm based on L-BFGS [70] to approximate the integral Hessian matrix and recover an accurate global model using strategies such as warm-up, periodic correction, and final tuning.

Descent-to-Delete [68] introduces a basic gradient descent algorithm that begins with the previous model and executes a limited number of gradient descent updates. This process ensures the model parameters remain in close Euclidean proximity to the optimal parameters. Subsequently, Gaussian noise is applied to the model parameters to ensure indistinguishability for any entity close to the optimal model. For high-dimensional data, it partitions the dataset, optimizes each part independently, and releases a perturbed average similar to FederatedAveraging [71].

BAERASER [69] applies gradient ascent-based unlearning to remove backdoors [72] in models. The process begins by identifying embedded trigger patterns. Once these triggers are discovered, BAERASER uses them to discard the contaminated memories through a gradient ascent-based machine unlearning method. The loss of trigger pattern unlearning is $F_U = -F_{CE}(\mathcal{M}(\mathbf{x}_b; \mathbf{w}_j), y_b)$, where F_{CE} is the prediction loss for a trigger pattern $\mathbf{x}_b$ with respect to the target label y_b, and $\mathbf{w}_j$ is the parameters of the target model at j_{th} unlearning iteration. The loss of trigger pattern unlearning is designed to maximize the cross-entropy loss between the model's prediction for a trigger pattern and the target label, thereby reducing the influence of the trigger pattern. To prevent the model's performance from dropping due to the unlearning process, BAERASER uses the validation data to maintain the memory of the target model over the normal data and a dynamic penalty mechanism to punish the over-unlearning of the memorizes unrelated to trigger patterns.

Comparisons and Discussions. Approximate unlearning based on gradient update can use cached information such as gradients and parameters to adapt models to small data changes rapidly. These methods offer more efficient alternatives compared to naive retraining but also have some limitations. Table IV summarizes and compares various approximate unlearning based on gradient update.

A key advantage is rapidly adapting models to small data changes with minimal computational expense. However, the techniques rely on assumptions such as strong convexity that may not hold for complex models [45], [68]. Techniques such as L-BFGS that approximate Hessians to speed up [45], [67]

may also break down for very high-dimensional models.

Another limitation is approximation errors can accumulate over multiple update rounds [68], resulting in less accurate recovered models. Strategies such as warm-up and periodic correction [45], [67] address this but introduce extra costs. The techniques also struggle with large data changes, as the gradient adjustments are insufficient to adequately adapt the model.

In summary, these gradient-based unlearning methods offer promising efficiency gains for data removal, but practical deployments must carefully validate assumptions and theoretical guarantees. Improvements to enhance robustness to approximation errors and large data changes could expand their applicability. Further research is needed to handle complex models, automated tuning, and formal privacy guarantees for widespread use in real-world machine learning systems.

E. Approximate Graph Unlearning

Graph-structured data brings unique challenges to machine unlearning due to the inherent dependencies between connected data points. Traditional machine unlearning methods designed for tabular or image data often fail to account for the complex interactions present in graph data. Recent studies have made important advancements in developing specialized techniques for approximate graph unlearning.

In this section, we introduce several papers that have made important contributions to the field of graph-based models. These papers propose methods to address the challenges of graph unlearning from different perspectives, which are crucial for a deep understanding of the current state of research in this field.

First, data interdependence is a key challenge in graph unlearning. Given a node in a graph as a removing target, it is necessary to remove its influence and its potential influence on multi-hop neighbors. To address this issue, Wu *et al.* [73] proposed a Graph Influence Function (GIF) to consider such structural influence of node/edge/feature on its neighbors. GIF estimates the parameter changes in response to ϵ-mass perturbations in the removed data by introducing an additional loss term related to the affected neighbors. GIF provides a way to explain the effects of unlearning node features. Cheng *et al.* [74] introduced two properties called Deleted Edge Consistency and Neighborhood Influence that aim to limit the impact of edge deletion to only the local neighborhood. Deleted Edge Consistency ensures that the deletion of an edge does not affect the representation of other edges in the same neighborhood, while Neighborhood Influence ensures that the deletion of an edge only affects its direct neighbors and not the entire graph. Their proposed method GNNDELETE incorporates these properties through a novel deletion operator and achieves strong performance on node and edge deletion tasks. Chien [75] aims to address three different types of data removal requests in graph unlearning: node feature unlearning, edge unlearning, and node unlearning. They derive theoretical guarantees for node/edge/feature deletion specifically for Simple Graph Convolutions and their generalized PageRank generalizations.

Second, most graph unlearning methods are designed for the transductive graph setting, where the graph is static and test graph information is available during training. However, many real-world graphs are dynamic, with new nodes and edges being continuously added. To address this, Wang *et al.* [76] proposed the GUIded InDuctivE Graph Unlearning framework (GUIDE) to realize graph unlearning for dynamic graphs. GUIDE includes fair and balanced guided graph partitioning, efficient subgraph repair, and similarity-based aggregation. Balanced partitioning ensures that the retraining time of each shard is similar, and subgraph repair and similarity-based aggregation reduce the side effects of graph partitioning, thereby improving model utility.

Third, it is challenging to achieve graph unlearning while maintaining model performance when the number of training data is limited. To address this, Pan *et al.* [77] proposed a nonlinear approximate graph unlearning method based on Graph Scattering Transform (GST). GST is stable under small perturbations in graph features and topologies, making it a robust method for graph data processing. Furthermore, GSTs are non-trainable, and all wavelet coefficients in GSTs are constructed analytically, making GST computationally more efficient and requiring less training data than GNNs.

Finally, in the realm of federated knowledge graph (KG) embedding learning, a significant concern is the effective handling of data heterogeneity and knowledge retention challenges. FedLU [78] uses mutual knowledge distillation to transfer local knowledge to the global model and absorb global knowledge into the local models. To achieve federated KG embedding unlearning, FedLU is inspired by the forgetting theories in cognitive neuroscience and adopts a two-step forgetting process of interference and decay. In the first retroactive interference step, FedLU performs hard and soft confusions, as the interference theory states that forgetting occurs when memories compete and interfere with other memories [79]. Then, in the passive decay step, FedLU suppresses the activation of forgotten knowledge, as the decay theory posits that memory traces gradually disappear and are eventually lost if not retrieved and rehearsed [80].

Comparisons and Discussions. Recent research on graph unlearning has extended machine unlearning techniques to accommodate the additional complexities of graph-structured data and node dependencies. However, there remain open challenges and opportunities for future work. The summary and comparison of approximate graph unlearning methods are in Table V.

A key insight is that graph unlearning requires specialized techniques that consider graph data's unique dependencies and constraints. Simple adaptations of existing unlearning methods are insufficient. For example, research [73] proposes graph-oriented influence functions by incorporating the loss of influenced neighbors, which outperforms conventional influence functions in GNN. An open question is how to develop a unified framework that can automatically determine the optimal graph unlearning strategy based on data properties.

Another limitation of current graph unlearning methods is scalability to large and dynamic graphs [76], [78]. Most techniques are only evaluated on small networks and may

TABLE V: Approximate Graph Unlearning Methods

Paper	Goal	Design Ideas	Strengths	Weaknesses
[73]	General unlearning strategy for GNNs	Graph influence functions considering neighbors' influence	• Applicable to different models • Supports various removal tasks • Closed-form solution	• Focus on the classification • Memory-intensive
[74]	Efficient unlearning for GNNs	• Deleted Edge Consistency • Neighborhood Influence	• Flexible removal operator applicable to any GNN • Supports various removal tasks	• Limited to transductive learning
[75]	Graph-structured data unlearning	• Update model parameters based on gradient difference	• Analyze for node/edge/feature unlearning • Strong theoretical guarantees	• Analysis limited to linear models • Loose worst-case bounds
[76]	Inductive graph unlearning	• Guided graph partitioning • Subgraph repairing • Similarity-based aggregation	• Repair subgraphs independently • Enables unlearning on evolving graphs	• Increased memory cost • Generalizing partition fairness needs exploration
[77]	Unlearning for GNNs with limited training data	Use GSTs for graph embeddings	• Nonlinear approximate graph unlearning • Theoretical guarantees	• Limited to classification • Bounds are loose
[78]	Unlearning for knowledge graphs in FL	• Mutual knowledge distillation • Retroactive interference & passive decay for unlearning	• Unlearning for heterogeneous federated KGs • Mutual knowledge distillation reduces bias in local and global models	• High computational cost • Applicability to other graph data needs exploration

not extend well to graphs with millions of nodes and edges that evolve over time. Developing efficient and incremental unlearning algorithms is an important direction for enabling real-world deployment.

An interesting area for further exploration is how to apply graph unlearning in advanced graph-based applications such as recommender systems [77], [78], node classification, and link prediction. Studying the influence of unlearning parts of a knowledge graph on downstream predictive tasks could provide insight into how much utility is retained.

There are also open questions around further characterizing the inherent tradeoffs between unlearning performance, privacy risks, and model utility [73], [74]. Quantifying these tradeoffs could help select optimal operating points based on application requirements.

Overall, the field of graph unlearning still has rich opportunities for impactful contributions through innovations in efficiency, scalability, flexibility, and rigorous characterization. By addressing the unique needs of graph data, future advances can expand the scope and applications of machine unlearning.

F. Approximate Unlearning based on Novel Techniques

Researchers have also developed novel approximate unlearning techniques that exploit unique model architectures or data characteristics.

Wang *et al.* [81] propose a method for selectively removing categories from trained Convolutional Neural Network (CNN) classification models in FL. This approach is based on the observation that different CNN channels contribute differently to image categories. They use Term Frequency Inverse Document Frequency (TF-IDF) to quantify the class discrimination of channels and prune highly discriminative channels of target categories to facilitate unlearning. When the federated unlearning process begins, the federated server notifies clients to calculate and upload local representations. The server then prunes discriminative channels and fine-tunes the model to regain accuracy, avoiding full retraining.

Izzo *et al.* [82] proposed the Projective Residual Update (PRU) for data removal from linear regression models. PRU computes the projection of the exact parameter update vector onto a specific low-dimensional subspace, with its computational cost scaling linearly with the data dimension, making it independent of the number of training data points. The PRU algorithm begins by creating synthetic data points and computing Leave-k-out (LKO) predictions. Next, the pseudoinverse of a matrix, composed of the outer product of the feature vectors of the removed data points, is computed. The model's loss at these synthetic points is then minimized by taking a gradient step, which also updates the model parameters. This characteristic makes PRU a scalable solution for handling large datasets, as the volume of training data does not compromise its efficiency.

ERM-KTP [40] is an interpretable knowledge-level machine unlearning method that removes target classes' influence for image classifiers. To achieve this, ERM-KTP employs an Entanglement-Reduced Mask (ERM) during training to separate and isolate class-specific knowledge. When unlearning is needed, Knowledge Transfer and Prohibition (KTP) selectively transfers non-target knowledge to a new model while prohibiting target knowledge transfer. The interpretable design enhances trust and transparency in the unlearning process.

Boundary Unlearning [83] aims to efficiently erase an entire class from a trained deep neural network by shifting the decision boundary instead of modifying parameters. It transfers attention from the high-dimensional parameter space to the decision space of the model, allowing rapid removal of the target class. Two methods are introduced: boundary shrink reassigns each removing data point to its nearest incorrect class and fine-tunes the model accordingly; boundary expanding temporarily maps all removing data to a new shadow class, fine-tunes the expanded model, and then prunes away the

shadow class.

Quark [84] aims to unlearn undesirable behaviors such as toxicity and repetition from large pre-trained language models using reinforcement learning techniques. It works by scoring model samples with a reward function and sorting into quantiles, appending special reward tokens denoting the quantile, retraining the model on each quantile conditioned on its token, with a KL penalty, and sampling at generation time with the best reward token to steer towards higher rewards.

V. Critical Issues of Machine Unlearning

A. Performance and Privacy Issues of Unlearning

Machine unlearning is an emerging technique in machine learning that provides the ability to selectively remove previously learned data from trained models [16]. As a supplementary tool to traditional ML models, machine unlearning enables efficient modifying, updating, and refining models. This subtractive capability facilitates use cases such as removing personal data for privacy, resisting poisoned data, and responding to new regulations.

However, machine unlearning also brings technical challenges and tradeoffs in ML system design. One challenge is the tradeoff of the efficiency of machine unlearning with model performance. Architectures such as SISA improve unlearning timeliness through ensemble models but can isolate data and hurt overall performance [85]. Well-designed unlearning algorithms that maintain the original ML model structure may still decrease model accuracy and even leads to catastrophic forgetting [43], [86]. The unlearning methods must be tailored to minimize negative impacts on the model. Another potential issue is that removing one data point may expose information about other data points, compromising privacy [42], [57]. It is necessary to combine machine unlearning with encryption techniques to mitigate such risks.

To tackle these challenges, advances in model architectures, dataset engineering, and infrastructure to support machine unlearning are required. For example, new model architectures could isolate data to limit negative impacts during unlearning. Synthetic dataset generation can create training data with built-in controls for later unlearning. With solutions across model, data, and system levels, machine unlearning can become a fundamental technique to construct more trustworthy, secure, and privacy-preserving ML systems.

B. Machine Unlearning and the Right to Be Forgotten

The right to be forgotten, allowing individuals to request personal data removal, is important for privacy protection. Although machine unlearning became popular partly because of its enabling the right to be forgotten, the relationship between machine unlearning and this right is nuanced. While machine unlearning provides a useful technical tool, it is neither necessary nor sufficient to guarantee the right to be forgotten.

First, machine unlearning is not strictly necessary for exercising the right to be forgotten, as other techniques, such as retraining models from scratch or on new datasets, can also fulfill data removing requirements. Machine unlearning provides a more computationally efficient method for forgetting, but other feasible approaches exist.

Second, machine unlearning alone is insufficient for comprehensively guaranteeing the right to be forgotten. Beyond just removing model parameters through machine unlearning algorithms, additional technical and legal steps are required to fully assert this right in practice, such as verifiable proof of unlearning, proof of data ownership, auditing for potential privacy leaks, and employing privacy-enhancing technologies.

Third, adapting machine unlearning for the right to be forgotten will also bring new threats. For example, malicious data owners could continuously initiate unlearning requests to hurt the model availability of model owners. Defending against such attacks remains an open challenge. Additionally, in data sharing, multiple parties share their data to train a common machine learning model. Malicious data owners could deliberately share low-quality data, unlearn it after obtaining the model, then retrain on their high-quality data to gain advantage [87].

Stronger legal and technical tools integration can better balance users' rights and company interests [88]. Aligning regulations and technical standards can enable compliant, minimized-cost unlearning algorithms [57]. Overall, machine unlearning provides useful techniques but must be implemented alongside other safeguards to encourage companies to act quickly to assert users' data protection rights.

C. Proof of Unlearning

Proof of unlearning is an important concept that provides auditable evidence that an ML model has properly unlearned certain training data points. While related to proof of learning [89], [90], proof of unlearning is more challenging as the adversary is the model owner themselves. Since they have access to the original training data and model, weaker forms of proof of learning can be circumvented [17]. Stronger cryptographic proofs are needed to prevent manipulation of the audit process.

Recent work [17] demonstrates that unlearning is best defined at the algorithmic level rather than by reasoning about model parameters. They show that model parameters can be identical even when trained with or without certain data points [91]. This means that inspecting parameters alone cannot prove whether unlearning occurred [17]. Other related work has proposed using techniques such as backdoors to detect compliance of a model owner with a data removal request [92], [93]. However, such techniques are only probabilistic and limited by data points.

To enable rigorous auditing, continued research on efficient cryptographic proofs for unlearning is important. Promising directions include succinct zero-knowledge proofs and trusted execution environments (TEEs) [94]. These can prove unlearning algorithms were properly executed without relying on indirect model observations. Key challenges include ensuring the transparency and reliability of proofs, minimizing verification time, and cost on the underlying machine learning pipelines.

By combining cryptography, trusted computing, and advances in machine learning, proof of unlearning can provide

robust auditing of unlearning claims. This emerging technique will be essential for accountability and compliance as the right to be forgotten is increasingly recognized. Ongoing research to develop practical proofs will support reliable validation of unlearning in complex, real-world systems.

D. Unlearning and Explainable AI

There is an intricate relationship between explainable artificial intelligence and machine unlearning. On the one hand, improving model explainability can facilitate the development of more effective unlearning algorithms. Techniques from explainable AI, such as generating feature importance scores and instance-based explanations [26], can help identify parts of the model most influenced by specific training data. Such insights can guide the design of unlearning methods that precisely remove the targeted information.

On the other hand, machine unlearning itself may benefit explainability. For example, by analyzing how explanations of model predictions change before and after unlearning certain data, one may gain insights into how the removed information initially influenced the model. Comparing explanations derived from the original model versus the updated model after unlearning can elucidate what memorization and unlearning entail in complex machine learning systems [95].

Overall, explainability and unlearning are mutually beneficial, and integrating the two leads to more transparent, trustworthy, and controllable AI systems. But there remain open challenges around developing rigorous quantitative evaluation protocols to assess the interplay between explainability and unlearning.

VI. Potential Research Directions

Machine unlearning is an emerging field with many open problems and opportunities. In this section, we discuss several promising directions for advancing machine unlearning research. By tackling existing methods' challenges and limitations, future work can significantly extend the scope, applicability, reliability, and flexibility of machine unlearning.

A. Unlearning for Diverse Data Structures

Existing studies have focused on exploring unlearning algorithms on set-structured and graph-structured training data. However, many other complex data structures such as text, speech, and multimedia are also increasingly used to train machine learning models, especially in fields such as natural language processing [2] and audio/video analysis [96]. Extending unlearning techniques to these diverse and complex data is an important next step but poses challenges. This requires handling the unique characteristics of different data types, such as timing and sequences in speech and language, spatial relationships in images, or hierarchical structures in knowledge graphs. Further, developing multimodal unlearning [97] that works across data combinations is also crucial for real-world usage. Tackling these data structure challenges can greatly expand the applicability of machine unlearning.

B. Unlearning for Non-convex Models

Non-convex neural network models such as CNNs and Recurrent Neural Networks (RNNs) have become prevalent in various deep learning applications. However, existing approximate unlearning research has mostly focused on convex models such as logistic regression. Extending efficient and effective unlearning algorithms to non-convex neural networks remains an important open challenge [48]. The key difficulties include dealing with non-convex optimization problems such as local optima and saddle points and providing theoretical guarantees on the unlearning process similar to those for convex cases. It also needs to deal with special structures in neural networks, such as activation functions and normalization layers [98]. To address these challenges, future research should aim to develop novel unlearning algorithms and analyses tailored to the characteristics of common neural network structures. This can significantly expand the applicability of machine unlearning in computer vision [99], natural language processing [2], speech recognition [100], and other areas that heavily rely on deep neural networks.

C. User-Specified Granularity of Unlearning

Most existing machine unlearning methods focus on instance-level removing, i.e., removing the influence of one training data point. However, users may need finer-grained control over what to remove from the model. For example, users may request to remove only certain sensitive regions of an image while retaining the rest or specific words in a text document that are no longer appropriate. An interesting research direction is to explore interactive and interpretable unlearning algorithms that allow users to specify the granularity of unlearning at a finer-grained level. Such algorithms may need to identify the semantic components of examples and their contributions to model predictions. It will greatly enhance the ability of unlearning techniques to meet user requirements.

D. Privacy Assurance for Unlearning

Most existing approximate unlearning algorithms rely on differential privacy to provide formal unlearning guarantees [48], [101]. However, differential privacy often uses a relatively relaxed privacy budget to balance privacy and utility [102]. The privacy guarantees it provides may be insufficient for scenarios with extremely high privacy demands [103], [104]. Therefore, an important research direction is to explore stronger privacy notions beyond differential privacy that can limit information disclosure more rigorously while not excessively sacrificing model utility. For example, exploring information theoretic approaches [105] that directly bound the amount of information about the removed data retained in the model after unlearning. This requires overcoming the difficulty of extracting information from models. This research direction can potentially promote the explainability and verifiability of machine learning algorithms [106].

E. Quantitative Evaluation Metrics

To compare different unlearning methods, it is crucial to develop quantitative metrics that can measure the degree

of influence removal for the removed data and the degree of influence retention for the remaining data [94], [107]. However, constructing such fine-grained evaluation metrics requires the ability to systematically analyze the memorization process of machine learning models on different data [107]. Advanced tools from information theory [108] and explainability research [109] are useful in this direction. Well-designed metrics will greatly promote the theoretical analysis of machine unlearning algorithms and guide the development and adoption of reliable unlearning techniques in practice.

VII. Conclusion

In this paper, we have presented a comprehensive overview of machine unlearning, an emerging technique that enables selective removing training data in machine learning models. We reviewed exact unlearning based on SISA, as well as approximate unlearning utilizing ideas from influence functions, re-optimization, gradient update, and other approaches. Our analysis reveals current limitations in computational complexity, providing strict privacy guarantees and limiting model utility. We suggest promising directions such as extending unlearning to diverse models and data structures, developing efficient and verifiable algorithms, establishing rigorous evaluation methods, and exploring connections with related fields. By addressing these challenges from different aspects, such as algorithms, systems, and regulations, machine unlearning can become an integral capability for trustworthy and adaptive AI.

References

[1] H. Zhao, J. Jia, and V. Koltun, "Exploring self-attention for image recognition," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 076–10 085.

[2] J. Hirschberg and C. D. Manning, "Advances in natural language processing," *Science*, vol. 349, no. 6245, pp. 261–266, 2015.

[3] S. Wu, F. Sun, W. Zhang, X. Xie, and B. Cui, "Graph neural networks in recommender systems: a survey," *ACM Computing Surveys*, vol. 55, no. 5, pp. 1–37, 2022.

[4] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[5] S. Fu, F. He, and D. Tao, "Knowledge removal in sampling-based bayesian inference," in *International Conference on Learning Representations*, 2021.

[6] A. K. Tarun, V. S. Chundawat, M. Mandal, and M. Kankanhalli, "Fast yet effective machine unlearning," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[7] M. Jegorova, C. Kaul, C. Mayor, A. Q. O'Neil, A. Weir, R. Murray-Smith, and S. A. Tsaftaris, "Survey: Leakage and privacy at inference time," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[8] H. Xu, T. Zhu*, L. Zhang, W. Zhou, and P. S. Yu, "Machine unlearning: A survey," *ACM Computing Surveys*, 2023.

[9] P. Voigt and A. Von dem Bussche, *The EU General Data Protection Regulation (GDPR). A Practical Guide*. Springer International Publishing, 2017.

[10] S. o. C. D. o. J. Office of the Attorney General, "California consumer privacy act (ccpa)," https://oag.ca.gov/privacy/ccpa, 2023.

[11] G. Wang, C. X. Dang, and Z. Zhou, "Measure contribution of participants in federated learning," in *2019 IEEE international conference on big data (Big Data)*. IEEE, 2019, pp. 2597–2604.

[12] H. Chang and R. Shokri, "On the privacy risks of algorithmic fairness," in *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2021, pp. 292–303.

[13] B. Biggio, B. Nelson, P. Laskov *et al.*, "Poisoning attacks against support vector machines," in *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*. ArXiv e-prints, 2012, pp. 1807–1814.

[14] J. Steinhardt, P. W. W. Koh, and P. S. Liang, "Certified defenses for data poisoning attacks," *Advances in neural information processing systems*, vol. 30, 2017.

[15] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM computing surveys (CSUR)*, vol. 46, no. 4, pp. 1–37, 2014.

[16] Y. Cao and J. Yang, "Towards making systems forget with machine unlearning," in *2015 IEEE symposium on security and privacy*. IEEE, 2015, pp. 463–480.

[17] A. Thudi, H. Jia, I. Shumailov, and N. Papernot, "On the necessity of auditable algorithmic definitions for machine unlearning," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 4007–4022.

[18] H. Yan, X. Li, Z. Guo, H. Li, F. Li, and X. Lin, "ARCANE: An Efficient Architecture for Exact Machine Unlearning," in *IJCAI International Joint Conference on Artificial Intelligence*, 2022, pp. 4006–4013.

[19] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*. Springer, 2006, vol. 4, no. 4.

[20] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[21] Z.-H. Zhou, *Ensemble methods: foundations and algorithms*. CRC press, 2012.

[22] L. I. Kuncheva and C. J. Whitaker, "Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy," *Machine learning*, vol. 51, pp. 181–207, 2003.

[23] L. Rokach, "Ensemble-based classifiers," *Artificial intelligence review*, vol. 33, pp. 1–39, 2010.

[24] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," *arXiv preprint arXiv:1702.08608*, 2017.

[25] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins *et al.*, "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai," *Information fusion*, vol. 58, pp. 82–115, 2020.

[26] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," in *International conference on machine learning*. PMLR, 2017, pp. 1885–1894.

[27] F. R. Hampel, "The influence curve and its role in robust estimation," *Journal of the american statistical association*, vol. 69, no. 346, pp. 383–393, 1974.

[28] S. Shan, A. N. Bhagoji, H. Zheng, and B. Y. Zhao, "Poison forensics: Traceback of data poisoning attacks in neural networks," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 3575–3592.

[29] B. Hui, Y. Yang, H. Yuan, P. Burlina, N. Z. Gong, and Y. Cao, "Practical blind membership inference attack via differential comparisons," in *Network and Distributed System Security Symposium (NDSS)*, 2021.

[30] H. Hu, Z. Salcic, L. Sun, G. Dobbie, P. S. Yu, and X. Zhang, "Membership inference attacks on machine learning: A survey," *ACM Computing Surveys (CSUR)*, vol. 54, no. 11s, pp. 1–37, 2022.

[31] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE symposium on security and privacy (SP)*. IEEE, 2017, pp. 3–18.

[32] M. Chen, Z. Zhang, T. Wang, M. Backes, M. Humbert, and Y. Zhang, "When machine unlearning jeopardizes privacy," in *Proceedings of the 2021 ACM SIGSAC conference on computer and communications security*, 2021, pp. 896–911.

[33] S. Garg, S. Goldwasser, and P. N. Vasudevan, "Formalizing data deletion in the context of the right to be forgotten," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2020, pp. 373–402.

[34] T. Bertram, E. Bursztein, S. Caro, H. Chao, R. C. Feman, P. Fleischer, A. Gustafsson, J. Hemerly, C. Hibbert, L. Invernizzi, L. K. Donnelly, J. Ketover, J. Laefer, P. Nicholas, Y. Niu, H. Obhi, D. Price, A. Strait, K. Thomas, and A. Verney, "Five years of the right to be forgotten," in *Proceedings of the ACM Conference on Computer and Communications Security*, 2019, pp. 959–972.

[35] V. S. Chundawat, A. K. Tarun, M. Mandal, and M. Kankanhalli, "Zero-shot machine unlearning," *IEEE Transactions on Information Forensics and Security*, 2023.

[36] Y. Cao, A. F. Yu, A. Aday, E. Stahl, J. Merwine, and J. Yang, "Efficient repair of polluted machine learning systems via causal unlearning," in *Proceedings of the 2018 on Asia conference on computer and communications security*, 2018, pp. 735–747.

[37] B. Mirzasoleiman, A. Karbasi, and A. Krause, "Deletion-robust submodular maximization: Data summarization with "the right to be

forgotten"," in *Proceddings of the 34th International Conference on Machine Learning, ICML 2017*, vol. 5, 2017, pp. 3780–3790.

[38] J. Serra, D. Suris, M. Miron, and A. Karatzoglou, "Overcoming catastrophic forgetting with hard attention to the task," in *International conference on machine learning*. PMLR, 2018, pp. 4548–4557.

[39] S. Schelter, S. Grafberger, and T. Dunning, "HedgeCut: Maintaining Randomised Trees for Low-Latency Machine Unlearning," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2021, pp. 1545–1557.

[40] S. Lin, X. Zhang, C. Chen, X. Chen, and W. Susilo, "Erm-ktp: Knowledge-level machine unlearning via knowledge transfer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 20 147–20 155.

[41] L. Bourtoule, V. Chandrasekaran, C. A. Choquette-Choo, H. Jia, A. Travers, B. Zhang, D. Lie, and N. Papernot, "Machine unlearning," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 141–159.

[42] N. Carlini, M. Jagielski, C. Zhang, N. Papernot, A. Terzis, and F. Tramer, "The privacy onion effect: Memorization is relative," *Advances in Neural Information Processing Systems*, vol. 35, pp. 13 263–13 276, 2022.

[43] J. Ye, Y. Fu, J. Song, X. Yang, S. Liu, X. Jin, M. Song, and X. Wang, "Learning with recoverable forgetting," in *European Conference on Computer Vision*. Springer, 2022, pp. 87–103.

[44] H. Jia, H. Chen, J. Guan, A. S. Shamsabadi, and N. Papernot, "A zest of lime: Towards architecture-independent model distances," in *International Conference on Learning Representations*, 2021.

[45] Y. Wu, E. Dobriban, and S. Davidson, "Deltagrad: Rapid retraining of machine learning models," in *International Conference on Machine Learning*. PMLR, 2020, pp. 10 355–10 366.

[46] A. Golatkar, A. Achille, and S. Soatto, "Eternal sunshine of the spotless net: Selective forgetting in deep networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9304–9312.

[47] S. Mercuri, R. Khraishi, R. Okhrati, D. Batra, C. Hamill, T. Ghasempour, and A. Nowlan, "An introduction to machine unlearning," *arXiv preprint arXiv:2209.00939*, 2022.

[48] C. Guo, T. Goldstein, A. Hannun, and L. Van Der Maaten, "Certified data removal from machine learning models," in *International Conference on Machine Learning*. PMLR, 2020, pp. 3832–3842.

[49] C. Dwork, "Differential privacy," in *Automata, Languages and Programming: 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II 33*. Springer, 2006, pp. 1–12.

[50] J. Brophy and D. Lowd, "Machine Unlearning for Random Forests," in *International Conference on Machine Learning*, sep 2020, pp. 1092–1104. [Online]. Available: https://proceedings.mlr.press/v139/brophy21a.html

[51] M. Chen, Z. Zhang, T. Wang, M. Backes, M. Humbert, and Y. Zhang, "Graph Unlearning," in *Proceedings of the ACM Conference on Computer and Communications Security*, 2022, pp. 499–513.

[52] C. Chen, F. Sun, M. Zhang, and B. Ding, "Recommendation Unlearning," in *WWW 2022 - Proceedings of the ACM Web Conference 2022*, 2022, pp. 2768–2777.

[53] A. A. Ginart, M. Y. Guan, G. Valiant, and J. Zou, "Making AI forget you: Data deletion in machine learning," in *Advances in Neural Information Processing Systems*, vol. 32, no. NeurIPS, 2019, pp. 1–14.

[54] N. Su and B. Li, "Asynchronous federated unlearning," in *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*. IEEE, 2023, pp. 1–10.

[55] Y. Liu, L. Xu, X. Yuan, C. Wang, and B. Li, "The right to be forgotten in federated learning: An efficient realization with rapid retraining," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 2022, pp. 1749–1758.

[56] A. Sekhari, J. Acharya, G. Kamath, and A. T. Suresh, "Remember what you want to forget: Algorithms for machine unlearning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 18 075–18 086, 2021.

[57] V. Suriyakumar and A. C. Wilson, "Algorithms that approximate data removal: New results and limitations," *Advances in Neural Information Processing Systems*, vol. 35, pp. 18 892–18 903, 2022.

[58] R. Mehta, S. Pal, V. Singh, and S. N. Ravi, "Deep unlearning via randomized conditionally independent hessians," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10 422–10 431.

[59] G. Wu, M. Hashemi, and C. Srinivasa, "Puma: Performance unchanged model augmentation for training data removal," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 8, 2022, pp. 8675–8682.

[60] R. Tanno, M. F Pradier, A. Nori, and Y. Li, "Repairing neural networks by leaving the right past behind," *Advances in Neural Information Processing Systems*, vol. 35, pp. 13 132–13 145, 2022.

[61] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, "Differentially private empirical risk minimization." *Journal of Machine Learning Research*, vol. 12, no. 3, 2011.

[62] A. Warnecke, L. Pirch, C. Wressnegger, and K. Rieck, "Machine unlearning of features and labels," in *30th Annual Network and Distributed System Security Symposium, NDSS 2023, San Diego, California, USA, February 27 - March 3, 2023*. The Internet Society, 2023. [Online]. Available: https://www.ndss-symposium.org/ndss-paper/machine-unlearning-of-features-and-labels/

[63] J. Martens, "New insights and perspectives on the natural gradient method," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 5776–5851, 2020.

[64] A. Golatkar, A. Achille, and S. Soatto, "Forgetting outside the box: Scrubbing deep networks of information accessible from input-output observations," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIX 16*. Springer, 2020, pp. 383–398.

[65] A. Golatkar, A. Achille, A. Ravichandran, M. Polito, and S. Soatto, "Mixed-privacy forgetting in deep networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 792–801.

[66] T. Shibata, G. Irie, D. Ikami, and Y. Mitsuzumi, "Learning with Selective Forgetting," in *IJCAI International Joint Conference on Artificial Intelligence*, 2021, pp. 989–996.

[67] X. Cao, J. Jia, Z. Zhang, and N. Z. Gong, "Fedrecover: Recovering from poisoning attacks in federated learning using historical information," in *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 2022, pp. 326–343.

[68] S. Neel, A. Roth, and S. Sharifi-Malvajerdi, "Descent-to-delete: Gradient-based methods for machine unlearning," in *Algorithmic Learning Theory*. PMLR, 2021, pp. 931–962.

[69] Y. Liu, M. Fan, C. Chen, X. Liu, Z. Ma, L. Wang, and J. Ma, "Backdoor defense with machine unlearning," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 2022, pp. 280–289.

[70] R. H. Byrd, J. Nocedal, and R. B. Schnabel, "Representations of quasi-newton matrices and their use in limited memory methods," *Mathematical Programming*, vol. 63, no. 1-3, pp. 129–156, 1994.

[71] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[72] H. Zhong, C. Liao, A. C. Squicciarini, S. Zhu, and D. Miller, "Backdoor embedding in convolutional neural network models via invisible perturbation," in *Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy*, 2020, pp. 97–108.

[73] J. Wu, Y. Yang, Y. Qian, Y. Sui, X. Wang, and X. He, "Gif: A general graph unlearning strategy via influence function," in *Proceedings of the ACM Web Conference 2023*, 2023, pp. 651–661.

[74] J. Cheng, G. Dasoulas, H. He, C. Agarwal, and M. Zitnik, "Gnndelete: A general strategy for unlearning in graph neural networks," in *The Eleventh International Conference on Learning Representations*, 2022.

[75] E. Chien, C. Pan, and O. Milenkovic, "Efficient model updates for approximate unlearning of graph-structured data," in *The Eleventh International Conference on Learning Representations*, 2022.

[76] C.-L. Wang, M. Huai, and D. Wang, "Inductive graph unlearning," *arXiv preprint arXiv:2304.03093*, 2023.

[77] C. Pan, E. Chien, and O. Milenkovic, "Unlearning graph classifiers with limited data resources," in *Proceedings of the ACM Web Conference 2023*, 2023, pp. 716–726.

[78] X. Zhu, G. Li, and W. Hu, "Heterogeneous federated knowledge graph embedding learning and unlearning," in *Proceedings of the ACM Web Conference 2023*, 2023, pp. 2444–2454.

[79] J. T. Wixted, "The role of retroactive interference and consolidation in everyday forgetting," in *Current Issues in Memory*. Routledge, 2021, pp. 117–143.

[80] O. Hardt, K. Nader, and L. Nadel, "Decay happens: the role of active forgetting in memory," *Trends in cognitive sciences*, vol. 17, no. 3, pp. 111–120, 2013.

[81] J. Wang, S. Guo, X. Xie, and H. Qi, "Federated unlearning via class-discriminative pruning," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 622–632.

[82] Z. Izzo, M. A. Smart, K. Chaudhuri, and J. Zou, "Approximate data deletion from machine learning models," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021, pp. 2008–2016.

[83] M. Chen, W. Gao, G. Liu, K. Peng, and C. Wang, "Boundary unlearning: Rapid forgetting of deep networks via shifting the decision boundary," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 7766–7775.

[84] X. Lu, S. Welleck, J. Hessel, L. Jiang, L. Qin, P. West, P. Ammanabrolu, and Y. Choi, "Quark: Controllable text generation with reinforced unlearning," *Advances in neural information processing systems*, vol. 35, pp. 27 591–27 609, 2022.

[85] Z. Ma, Y. Liu, X. Liu, J. Liu, J. Ma, and K. Ren, "Learn to forget: Machine unlearning via neuron masking," *IEEE Transactions on Dependable and Secure Computing*, 2022.

[86] Q. P. Nguyen, R. Oikawa, D. M. Divakaran, M. C. Chan, and B. K. H. Low, "Markov chain monte carlo-based machine unlearning: Unlearning what needs to be forgotten," in *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, 2022, pp. 351–363.

[87] Q. P. Nguyen, B. K. H. Low, and P. Jaillet, "Variational bayesian unlearning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 16 025–16 036, 2020.

[88] E. F. Villaronga, P. Kieseberg, and T. Li, "Humans forget, machines remember: Artificial intelligence and the right to be forgotten," *Computer Law & Security Review*, vol. 34, no. 2, pp. 304–313, 2018.

[89] H. Jia, M. Yaghini, C. A. Choquette-Choo, N. Dullerud, A. Thudi, V. Chandrasekaran, and N. Papernot, "Proof-of-learning: Definitions and practice," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 1039–1056.

[90] R. Zhang, J. Liu, Y. Ding, Z. Wang, Q. Wu, and K. Ren, ""adversarial examples" for proof-of-learning," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 1408–1422.

[91] I. Shumailov, Z. Shumaylov, D. Kazhdan, Y. Zhao, N. Papernot, M. A. Erdogdu, and R. J. Anderson, "Manipulating sgd with data ordering attacks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 18 021–18 032, 2021.

[92] X. Gao, X. Ma, J. Wang, Y. Sun, B. Li, S. Ji, P. Cheng, and J. Chen, "Verifi: Towards verifiable federated unlearning," *arXiv preprint arXiv:2205.12709*, 2022.

[93] D. M. Sommer, L. Song, S. Wagh, and P. Mittal, "Athena: Probabilistic verification of machine unlearning," *Proceedings on Privacy Enhancing Technologies*, vol. 3, pp. 268–290, 2022.

[94] J. Weng, S. Yao, Y. Du, J. Huang, J. Weng, and C. Wang, "Proof of unlearning: Definitions and instantiation," *arXiv preprint arXiv:2210.11334*, 2022.

[95] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.

[96] N. Takahashi, M. Gygli, and L. Van Gool, "Aenet: Learning deep audio features for video analysis," *IEEE Transactions on Multimedia*, vol. 20, no. 3, pp. 513–524, 2017.

[97] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, "Multimodal deep learning," in *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011, pp. 689–696.

[98] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. pmlr, 2015, pp. 448–456.

[99] A. Ioannidou, E. Chatzilari, S. Nikolopoulos, and I. Kompatsiaris, "Deep learning advances in computer vision with 3d data: A survey," *ACM computing surveys (CSUR)*, vol. 50, no. 2, pp. 1–38, 2017.

[100] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE international conference on acoustics, speech and signal processing*. Ieee, 2013, pp. 6645–6649.

[101] V. Gupta, C. Jung, S. Neel, A. Roth, S. Sharifi-Malvajerdi, and C. Waites, "Adaptive machine unlearning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 16 319–16 330, 2021.

[102] M. Jagielski, J. Ullman, and A. Oprea, "Auditing differentially private machine learning: How private is private sgd?" *Advances in Neural Information Processing Systems*, vol. 33, pp. 22 205–22 216, 2020.

[103] B. Jayaraman and D. Evans, "Evaluating differentially private machine learning in practice," in *28th USENIX Security Symposium (USENIX Security 19)*, 2019, pp. 1895–1912.

[104] X. Hu, M. Yuan, J. Yao, Y. Deng, L. Chen, Q. Yang, H. Guan, and J. Zeng, "Differential privacy in telco big data platform," *Proceedings of the VLDB Endowment*, vol. 8, no. 12, pp. 1692–1703, 2015.

[105] C. E. Shannon, "A mathematical theory of communication," *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[106] T. Eisenhofer, D. Riepel, V. Chandrasekaran, E. Ghosh, O. Ohrimenko, and N. Papernot, "Verifiable and provably secure machine unlearning," *arXiv preprint arXiv:2210.09126*, 2022.

[107] A. Thudi, G. Deza, V. Chandrasekaran, and N. Papernot, "Unrolling sgd: Understanding factors influencing machine unlearning," in *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2022, pp. 303–319.

[108] N. Tishby, F. C. Pereira, and W. Bialek, "The information bottleneck method," *arXiv preprint physics/0004057*, 2000.

[109] C. Molnar, *Interpretable machine learning*. Lulu. com, 2020.