# A Survey on Federated Unlearning: Challenges and Opportunities

Hyejun Jeong, Shiqing Ma, Amir Houmansadr

Abstract—Federated learning (FL), introduced in 2017, facilitates collaborative learning between non-trusting parties with no need for participants to explicitly share raw data. This allows training a model on different private sets of user data while respecting privacy regulations such as GDPR and CPRA. However, emerging privacy requirements may mandate model owners to be able to forget learned data, e.g., when requested by data owners or law enforcement. This has given birth to an active field of research called machine unlearning. In the context of FL, many techniques developed for unlearning in centralized settings are not trivially applicable! This is due to the unique differences between centralized and distributed learning settings, specifically interactivity, stochasticity, heterogeneity, and limited accessibility in FL. In response, a recent line of work has focused on developing unlearning mechanisms tailored to FL.

This survey paper aims to take a deep look at the *federated unlearning* literature, with the goal of identifying research trends and challenges in this emerging field. By carefully categorizing papers published on FL unlearning (since 2020), we aim to pinpoint the unique complexities of federated unlearning, highlighting limitations on directly applying centralized unlearning methods. We compare existing methods regarding influence removal and performance recovery, compare their threat models and assumptions, and discuss their implications and limitations. We additionally analyze the experimental setup from various perspectives, including data heterogeneity and its simulation, datasets used for demonstration, and evaluation metrics. Our work aims to offer insights and suggestions for future research on federated unlearning.

Index Terms—Federated unlearning, federated learning, machine unlearning.

## I. Introduction

THE "Right to be Forgotten" (RTBF) has gained attention upon its official recognition in 2014 <sup>1</sup>, aligning closely with privacy preservation mandates. The General Data Protection Regulation (GDPR) further stipulated the right to erasure <sup>2</sup>, empowering individuals to request personal data removal. Individuals may want their data to be forgotten from a trained model for various reasons, including privacy, security, or usability concerns. If they no longer consent to the privacy policy or terms of service, the data providers should be able to request the elimination of their explicit information as well as the influence on the model. From the model owner's perspective, if they found a model is trained on compromised or faulty data, such as backdoor poisoned data samples, they would also want to remove this data to improve security and utility [1]. This has motivated an active line of work referred to as Machine Unlearning (MU), aiming to remove (sensitive or faulty) data from trained ML models without retraining.

The trivial way to unlearn is to retrain the model from scratch, excluding the data to be removed. However, this is not only expensive in terms of overhead in time, memory, and resource consumption, but it may also not be feasible in various practical scenarios (e.g., the data to be unlearned are not open to the public). Therefore, the community has developed unlearning mechanisms to efficiently erase the requested data and its influence on the model while minimally impacting the model's performance [2]. Unlearning steps begin during or after training or convergence and continue until evaluation metrics are satisfied, generating *unlearned model* [3].

Unlearning in Federated Learning (FL): FL [4] is a distributed machine learning framework that emerged in 2017 to comply with stringent privacy policies. It enables multiple clients to collaboratively train a model while maintaining their data locally. This paradigm preserves data privacy by design, reducing the risk of privacy breaches from eavesdropping or misuse by data-collecting entities. In FL, a server initializes a global model and distributes the parameters to clients. Each client then trains a local model with private data and sends the trained model to the server, which aggregates the local models to update the global model. This iterative process continues until a stopping criterion is met.

In the context of FL, unlearning techniques developed for centralized settings are not trivially applicable! This is due to the unique differences between centralized and distributed learning, in particular, interactivity, stochasticity, heterogeneity, and limited accessibility. Once a model is trained in a federated manner, unlearning must also be performed in a federated setting. When a client requests to unlearn its data, it may perform centralized unlearning locally and send the updates to the server for aggregation. However, due to the interactivity inherent in FL, the unlearning effect can be trivial if the server simply averages all the local models. To address this, the target client, server, and remaining clients must collaborate to achieve proper unlearning, even with no access to the raw target data. To this end, recent works have focused on unlearning mechanisms tailored to FL [5].

## A. Contributions of This Survey

This paper aims to provide a comprehensive view of the state of Federated Unlearning (FU) literature, offering insights and recommendations for future research on this topic. In summary, our contribution can be summarized as follows:

 We highlight the unique complexities of unlearning in the federated setting, demonstrating the need for tailored unlearning mechanisms for FL (Section IV).

<sup>&</sup>lt;sup>1</sup>https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A62012CJ0131

<sup>&</sup>lt;sup>2</sup>https://eur-lex.europa.eu/eli/reg/2016/679/oj

TABLE I: Comparisons to other surveys. Full, partial, and no coverage are represented by  $\bullet$ ,  $\bullet$ , and  $\bigcirc$ , respectively.

Ref.	Year		FU				Unl	learı	ning			Ins	ight
		Federated context	Complexity specific to FL	Non-IID Data & simulation	Taxonomy	Who unlearns	Implication	Influence removal	Performance recovery	Limitation	Evaluation metrics	Dataset statistics	Research direction
[3]	2022	$  \Phi  $	•	0		0	•	•	0	•	•	•	•
[6]	2023	lack O	lacksquare	$\bigcirc$		$\bigcirc$	$\bigcirc$		$\bigcirc$				
[7]	2023			$\bigcirc$		$\circ$	$\circ$		$\circ$	$\circ$			
[8]	2023		$\circ$	$\circ$			$\circ$	lacksquare	$\circ$	$\circ$	$\bigcirc$		$\circ$
[9]	2023			$\bigcirc$		$\bigcirc$		lacksquare	$\circ$	$\bigcirc$			
[10]	2023			lacksquare			Ō		lacksquare				
[11]	2024	lack O	$\bigcirc$	$\bigcirc$		$\bigcirc$	Ō		Ō	$\bigcirc$			
[12]	2024						Ō		Ō			0	
Ours	2025												

- We identify and compare the assumptions made in existing FU literature. This includes the entities undertaking unlearning, data distribution specifications, dataset usage, models utilized, and aggregation methods employed, and the research implications (Section V).
- We compare unlearning targets in different works (Section III), categorize existing FU techniques regarding influence removal and performance recovery, and discuss their limitations (Section VI). We also compare the evaluation objectives and metrics used (Section VII).
- We provide suggestions for future research on FU based on lessons and insights from our investigation of the existing FU literature (Section VIII).

## B. Comparison to Other Surveys in Unlearning

In Table I we compare several recent surveys on Machine Unlearning [3, 6, 11] and Federated Unlearning [7–10] regarding various dimensions. As can be seen, while some surveys have considered unlearning in the federated context, not all addressed the unique complexities of FU, often overlooking factors such as data heterogeneity in FL. Furthermore, because FU involves multiple parties with varying knowledge and capabilities, disclosing raw original and unlearning model parameters to all participants could potentially facilitate adversaries in inferring the removed information. Thus, identifying the parties involved in unlearning and implementing robust safeguards is one of the necessary aspects to be considered, although only a few surveys [8, 10, 12] discussed it.

Compared to the standard machine unlearning, federated unlearning is in its infancy. By searching on Google Scholar, we try to identify works related to the keywords "unlearning" and "federated" or "distributed" at the same time. Figure 1 summarizes the number of FU papers published each year until 2024 (i.e., 85 by 2024, 94 by March 2025). We only include, for example, LLM unlearning literature, only demonstrated on the federated context, as this work focuses on federated

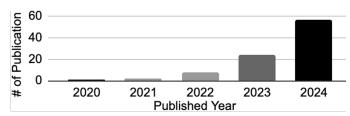


Fig. 1: Number of Federated Unlearning Publications.

unlearning. Although not exhaustive, the table reflects the exponentially increasing attention to FU, motivating our survey on identifying research gaps and challenges for future research.

1) Comparison to a concurrent survey [10]: The work most closely aligned with ours is [10], primarily reviewing existing FU methods and evaluation metrics. As our survey aims ultimately to provide insights for future work, we identify room for improvement in addition to analyzing existing literature in diverse dimensions. Specifically, our work delves deep into the practicality of each literature, including assumptions on data distribution and its simulation, along with dataset usage and research implications. We extract key takeaways from each part while exploring various concepts, properties, and methodologies. At the end of the paper, we provide valuable insights and potential directions based on the takeaways for readers' clear understanding.

## II. PRELIMINARIES

We provide preliminaries on FU, including definitions, brief introductions to FL and MU, and the inherent complexities.

## A. Federated Learning

As illustrated in Figure 2, FL [4] trains a model in a privacy-preserving way by distributing model updates to the participating clients instead of collecting sensitive data in a centralized location. The basic FL workflow is as follows:

- ① A central server initializes a global model  $w^0$ .
- ② The server advertises the global model  $w^t$  and selects  $K = C \times N$  clients, a random fraction C of clients out of a total of N clients that will participate in the current training round.
- ③ Each K client computes local updates  $g_k^t$  (as Eq. 2) with its training dataset  $\mathcal{D}_k$  based on the current global model  $w^t$ . Then, they send the updates  $g_k^t$  to the server.

$$F_k(w^t) = \frac{1}{D_k} \sum_{d \in \mathcal{D}_k} f_d(w_k^t) \tag{1}$$

$$g_k = \nabla F_k(w_t) \tag{2}$$

where  $\mathcal{D}_k$  is the set of data points with a size of  $D_k$  a client k has,  $w_k^t$  is the local model weight at round t,  $f_d$  is the loss of the prediction on sample d, and  $F_k$  is the average loss over the set of data points.

4 The server aggregates K received updates and creates an updated global model  $w^{t+1}$  (as Eq. 3).

$$w^{t+1} \leftarrow w^t - \eta \frac{1}{K} \sum_{k=1}^K g_k \tag{3}$$

where  $\eta$  is the learning rate.

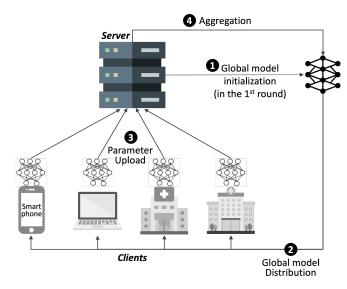


Fig. 2: FL Training Workflow.

2) to 4) are repeated until the pre-defined stopping criterion is met. FL algorithms are named differently based on how they collect and process updates to create the global model. FedAvg [4], for instance, is a foundational FL algorithm that averages the local model parameters. Whereas FedBuff [13] is a more recent method designed to reduce communication overhead and handle heterogeneous data by buffering updates. **Components.** A global server and multiple local clients comprise an FL process. The server, assumed to be equipped with more powerful computation and memory resources, is responsible for aggregating local clients' model parameters to update the global model. Nonetheless, as it cannot access raw data, its knowledge is limited to the model parameters but not training data. Similarly, local clients can access their own training dataset, not each other's. Key terminologies related to FL are summarized in Table VII in Section A.

## B. Machine Unlearning

MU aims to remove specific samples, classes, or features from a trained model upon request. A naive approach is to retrain the model from scratch, excluding the data to forget [1]. Nonetheless, this is often impractical due to its overhead, especially when the model has been trained with a huge dataset. As such, efficient unlearning methods, faster than *retraining*, have been explored, effectively forgetting the data while maintaining performance on remaining data. The objective is to make the unlearned model perform as similarly as possible to the retrained one. MU methods are broadly categorized into data-driven [3, 6] approaches including data partition [14], obfuscation [1, 15], or augmentation [16], and model manipulation [6, 17] approaches include model shifting, pruning, and replacement.

The evaluation metrics are various [3]. Broadly categorizing unlearning effectiveness metrics, there are performance-based (accuracy or error rates), distance-based (KL divergence, L2 distance, or cosine similarity), and attack-based (attack success rate of backdoor or membership inference attacks) metrics.

Many of these metrics overlap with those used for FU. Detailed evaluation metrics are explained in later sections.

Complexities in Machine Unlearning. The primary challenge in MU is to maintain the model's performance on remaining data while completely forgetting specific knowledge. Erasing the sample (*target removal*) is straightforward, but eliminating its influence is uneasy. The stochastic nature of training with randomly ordered batches, incremental nature of training, and catastrophic forgetting leads to significant performance degradation on the remaining data [3].

#### III. Introducing Federated Unlearning

Figure 3 depicts the overall FU flow. During or after training an FL model, either a server or clients request to remove certain target information. If the server initiates unlearning, for instance, to remove low-quality data, it additionally takes a step to identify the low-performing client to be deleted. Otherwise, the target samples or clients are removed from the learning process upon request.

What is Unlearned? Unlearning target refers to the data or information we want the model to forget. Upon request, the target(s) are removed from the training set (target removal). Although some methods [18–26] unlearn knowledge in multiple granularity, all client, class, and samples or two of them, most of the works focus on unlearning (a) client(s) because of the introduction of it in the federated context, unlike the centralized setting. We explain the different granularities of knowledge that are unlearned.

**Client** removal involves erasing all data owned by specific clients from the model [5, 21, 23, 27–57]. It overlaps with sample removal if all data owned by a specific client are subject to be removed [2, 58–64], and class removal when each client has a distinct class of samples [65, 66].

Class removal aims at removing all samples labeled as the target class. Think of label flipping or backdoor attacks; compromised clients hold trigger-injected samples labeled as a certain class (e.g., "9") while other clients hold normal data. Here, the target to unlearn would be the class 9 [35, 67–69]. Depending on sample distribution across clients, it can also be treated as a client removal problem [26, 70].

**Sample** removal involves eliminating particular data instances (e.g., removing mistakenly uploaded or private photos and their influence) [71–73]. It includes sanitizing the training dataset by removing polluted samples, such as compromised [34, 61, 67, 74–77] or outdated samples [6, 7, 78].

**Feature** removal is forgetting specific sensitivity in training data [79]. It can be used to improve model generalizability; for example, features could represent data heterogeneity, as seen in [80], that removed the scanner bias from medical images collected from different sites. It also overlaps with sample removal when eliminating samples with poison-indicating features and is analogous to client removal in vertical or cross-silo FL that each client has distinct features [81–85].

## IV. Unique Challenges of Federated Unlearning

The federated context brings unique complexities to FU, in addition to those in centralized unlearning, described in

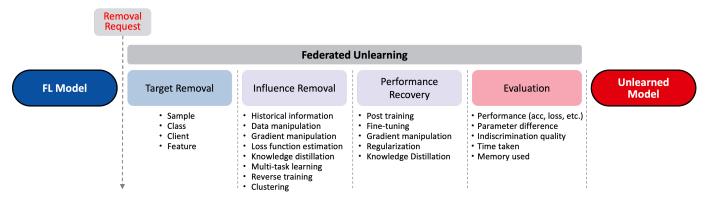


Fig. 3: Federated Unlearning Workflow. The server or clients initiate target removal during or after the FL process. The unlearner excludes the target, erases its contribution, and recovers performance. The requesting client verifies the proper elimination using evaluation metrics, generating an unlearned model.

Section II-B. In this section, we explore why MU methods cannot be directly applied to FU and investigate the inherent difficulties involved. Note that the complexities are not orthogonal nor mutually exclusive but usually related; to address one concern, the other might also need to be resolved. It makes complete FU much more complex to achieve.

Interactive training embeds information in a model [5, 7]. The subsequent training rounds accumulate all the information propagated throughout global and local models [27]. In addition to the information forwarding coupling in parameter updates, interactivity further adds many more difficulties. FL interactively trains a model by iteratively aggregating local models on the server, diminishing the impact of a local model upon aggregation. It then means that removing the information and influence from only one local model, analogous to that in MU, is no longer feasible in the federated context as it soon becomes ineffective after it is aggregated.

Information isolation characterizes FL that allows privacy-preserving training by keeping data local to clients, unlike centralized (un)learning, where a single entity has access to all data. FU involves three parties with varying levels of data accessibility: the server, target clients, and remaining clients. When the server or remaining clients perform unlearning, it is different from centralized unlearning and more complicated as they do not have access to raw data. Moreover, unlearning methods utilizing raw data (e.g., data partition [14], augmentation [16], or influence [17]) become inapplicable due to limited access to local data from the server or other clients [3, 7].

**Non-IID** data across clients, one of the famous open problems in FL [86], adds difficulty to FU. In contrast to centralized ML, each FL client separately trains a local model based on its training data that could vary in distribution. This heterogeneity results in some clients holding unique or skewed data distributions, causing their updates to disproportionately impact specific model parameters. It complicates isolating and eliminating the influence of target data, as parameter updates reflect a blended contribution from multiple clients.

**Stochastic client selection** introduces an additional layer of uncertainty, as the server receives updates from arbitrarily selected clients in each training round [7, 27]. This extends the stochasticity seen in MU with randomly ordered batch [3].

Since a client's impact is spread across non-contiguous rounds, pinpointing when and how to modify model parameters becomes challenging. Moreover, on-and-off client participation causes delayed propagation of updates, meaning the unlearned impact can persist in subsequent rounds, even when they are not actively participating. While frequent checkpointing or rollback can address this, it is highly resource-intensive, especially if the client participated early in training.

TABLE II: Accessible knowledge depending on who unlearns. "Global," "Own local," "All local," and "Raw data" refer to the global model updates, own local model updates, all local model updates, and raw data to forget, respectively.

Unlearner	Global	Own local	All local	Raw data
Server	✓		✓	
Target client	$\checkmark$	$\checkmark$		$\checkmark$
Remaining clients	$\checkmark$	$\checkmark$		

T-1. Takeaways on Unique Challenges of FU. Unlike MU, which unlearns information within accessible raw data in a central location, FU cannot solely rely on the target client due to the diminishing impact of individual clients upon aggregation. Isolated information, limited accessibility to data and its distribution, and extra uncertainty stemming from client selection introduce additional challenges to FU, as not all participants consistently have access to raw data.

#### V. Comparison of Existing System Models in FU

We overview and compare the system models used in the existing FU literature from various dimensions.

## A. Who Unlearns?

"Who unlearns" differentiates FU from MU and FL. In the federated context, the entities performing unlearning have knowledge and capabilities that depend on who they are. Table II summarizes the knowledge available to each unlearner. Each client has its local and global model updates but no updates from the others. Thus, if only the remaining clients unlearn, they must rely on the global model updates they have received. When the target client performs unlearning, it can additionally leverage its raw training data. On the other hand, if the server performs unlearning, it has access to the global model and updates from all clients' models. As indicated in the "Unlearner" column of Table III, most works chose to perform unlearning on the target clients, given their access to raw data to unlearn. The server, accessible to all local updates and identifying target clients, often takes charge of unlearning. Nonetheless, in scenarios where target clients request removal and leave, remaining clients often cooperate with the server.

**T-2. Takeaways on Who Unlearns.** The clients' flexibility to dynamically join and leave poses a risk of malicious clients joining FU. Furthermore, if the target client has left the system and/or the unlearning performer is not the target client, unlearning would not be properly completed, leaving residuals to the early-leaving client. Consequently, it could leave room for potential exploitation by compromised parties, who unlearn in what way matters to assure complete removal, distinct from FL.

## B. What Data Distribution?

Another crucial consideration is "under what data distribution," which is how data are distributed across clients. Data distribution and its simulation are summarized in the column "Data Dist." and "NIID sim." of Table IV. Heterogeneous data, a more practical assumption in the distributed nature, often interferes with FL models from convergence [110]. Similarly, an FU method designed for and demonstrated with IID data only would be highly unlikely to work as expected in practice with real-world data. Some work [28, 30, 78] naturally achieved it using real-world datasets, such as FEMNIST, Tiny-Shakespeare, or MovieLens-100k. The majority of the authors adopted prior probability shifts, as well as covariate shifts and concept drift. Below is a brief introduction to the three types and their implementation. For a more detailed description of non-IID data, refer to Section 3.1 in [86].

**Prior probability shift**, a variation in label distributions across clients, is often simulated using three primary methods:

- Dirichlet Distribution: This method quantifies the degree of non-IIDness using a concentration parameter, α (within the interval (0, ∞)<sup>4</sup>. As α → ∞, it mimics IID, while α → 0 imitates a higher degree of non-IID. Widely recognized for its ability to emulate real-world scenarios, many works employed this approach [18, 30, 34, 40, 51, 53, 55, 56, 58, 62, 64, 69, 70, 76, 79, 97, 100, 112].
- Fang's Method [113]: In this unique approach, clients are divided into L groups, where L denotes the number of classes in a dataset. Training instances with label l are assigned to the l-th group with probability p and to any other group with a probability of  $\frac{1-p}{L-1}$ . This method is adopted by [2, 107].

TABLE III: Unlearner. Svg, Tgt, and Rmn refer to the server, target, and remaining clients, respectively.

	T I.	alaam		ı	T T	nlaan	
Ref.	Svr	nlearı Tgt	Rmn	Ref.	Svr	nlear Tgt	Rmn
D EDECO11	3VI	ıgı	KIIIII	2521.5601	SVI	1gt	KIIIII
RevFRF[81]	•	_	_	2F2L[60]		•	
Exact-Fun[87]		•	•	Liu et al.[27]	_	•	
FedUHB[50]	_		•	Xie et al.[48]	•	_	
FATS[88]	•			Li et al.[43]	•	•	
R2S[46]	•		_	BadUnlearn[56]	•	_	
Zuo et al.[21]	_		•	Appro-Fun[52]		•	
FedADP[32]	•			FedU[42]		•	_
Shao et al.[89]	•		_	FedME2[71]		•	•
Wang et al.[90]	_		•	Alam et al.[75]		•	
FedRecover[2]	•		•	BFU[74]		•	_
Wu et al.[34]	•			FedHarmony[80]	_	•	•
Fu et al.[91]	•			Chen et al.[92]	•		
FedRecovery[29]	•			Goldfish[93]	•	_	_
CFRÚ[57]	•			FedLU[94]	•	•	•
FedADP[46]				FedAF[61] FedQUIT[51]		•	
MetaFul[64]	-			CKGD [55]	_	•	
Deng et al.[82]	3				•		
Starfish[95]	•• <sup>3</sup>			SFU[73]	•		_
Crab[96]	•		_	HDUS[66]		_	•
FedEraser[5]	_	_	•	FCU[40]	_	•	
FRU[59]	•	•	•	Ferrari[79]	•	•	_
SIFU[97]	•		•	Imba-ULRc[45]	•		•
FedRemover[98]	•			Xu et al.[68]	_		•
FedUnlearn[99]	•	_		TrustChain[21]	•		_
FedUNRAN[49]		•	_	FUSED[25]	•	_	•
FUCRT[69]		•	•	EWC-SGA[19] SFU[76]	_	•	_
FedMUA[100] Pan et al.[85]	_	•		Halimi et al.[101]	•	•	•
SecForget[102]	•	_				•	_
FFMU[103]		-		QuickDrop[70] LMR[67]	_	•	•
FedFilter[78]	_	•		Wang et al.[77]	•	-	
UKRL[31]		_		Han et al.[84]		-	
MoDe[62]	_			BlockFUL[22]		-	
ConDa[33]		•		FedOSD[41]	_	•	
RobustFU[35]		_	_	F2ul[38]	•		
FRAMU[104]	•	-		VFU[105]			•
VeriFi[58]	•		•	forgetSVGD[65]	•	•	
Lin et al.[106]				CforgetSVGD[26]	•	•	•
FC[28]				KNOT[30]	•		•
Wang et al.[107]	ě	•	•	Lin et al.[108]	•	•	
SecureCut[83]	•	ě	•	Lin et al.[109]	•		•
FAST[63]	•	•		Liu et al.[36]	•		•
ElBedoui et al.[72]	-	•		Hier-FUN[44]		•	-
Fast-FedUL[37]	•	•		FedUMP[47]	•	•	
NoT[53]	ě			k-IPfedAvg[23]	ě		
Wang et al.[24]	•			FedAU[18]	•	•	

<sup>&</sup>lt;sup>4</sup> Uses two servers for two-party computation, for security

• Unique Class Assignment: [26, 41, 65, 66] assign unique classes to each client such that each client exclusively holds data samples of one or two classes out of ten.

Covariate shift, or feature distribution skew, occurs when each client has different features of data instances with the same label shared across clients. An example is the dataset that FedHarmony [80] used. They wanted to unlearn site-specific bias from medical images so that the model could focus solely on biological features.

Concept drift refers to the changing input data distribution over time. When a sample is removed from the training dataset, the distribution shifts from its original state. With each subsequent sample removal, the distribution changes again, transitioning to a different state. FRAMU [104] views unlearning as a continual learning process from distinct training datasets after each removal, so it adapts to these dynamic and evolving distributions.

**T-3. Takeaways on Data Distribution.** More works assumed heterogeneous data in the federated context, although some undisclosed their specific simulation

<sup>&</sup>lt;sup>4</sup>For a visual representation, refer to Figure 4 in [111]

methods. The primary choice is employing Dirichlet distribution because of its recognized ability to mimic real-world datasets. However, as what value reflects the real world the most remains unclear,  $\alpha$ , controlling degree of non-IID, varies by a large margin across literature, limiting fair comparison across works.

## C. On What Dataset?

Existing FU methods demonstrated their effectiveness in diverse types of training data and for various tasks, such as image classification, object detection, regression, sentiment analysis, recommendation, and clustering. Nonetheless, as depicted in Figure 4, about 53 out of 94 studies demonstrated its effectiveness primarily on simple image datasets like MNIST and CIFAR10 <sup>6</sup> while fewer works have experimented on more complex image datasets such as CelebA <sup>7</sup>, CIFAR100 or EMNIST<sup>8</sup>. In total, image datasets were used 209 times, 80% of the entire usage, and multi-modal datasets were used twice in a single work, [28].

As a vast number of works have proven their unlearning capabilities with simple image datasets, recent works are expanding their experiments with either more complex images or text or both datasets. In particular, mostly from 2024, proposed FU works demonstrated their capabilities with image and text datasets (e.g., CIFAR100 and Shakespear datasets), showing their method is not limited to a specific type of dataset [30, 32, 34, 46, 58, 73, 79, 88, 108].

**T-4.** Takeaways on Dataset Usage. Like the early stages of FL, FU is yet predominantly focused on vision tasks with simple image datasets. Nonetheless, the practicality of FU should not be limited to basic datasets like MNIST. For FU to fully leverage the advantages offered by AI and FL, it should extend its scope to include diverse datasets for more sensitive tasks beyond vision tasks on images.

# D. Configurations

As detailed in Table IV, images are the most used data type for vision tasks in FU, so convolution networks were employed the most. Model architectures vary from shallower designs (2- to 4-layer CNN) to more complex structures, including LeNet, VGG, ResNet, DenseNet, and MobileNet. Most recent works in 2025 [53, 112] include vision transformers ViT-B/16 and SimpleViT, respectively. For some language tasks, we can observe the use of LSTM in [73, 88], BERT in [79], and GPTs in [21, 30, 108]. In a different domain, MetaFul [64] utilized LSTM to train a model on the video dataset. FRU [59] and CFRU [57], a federated unlearning method

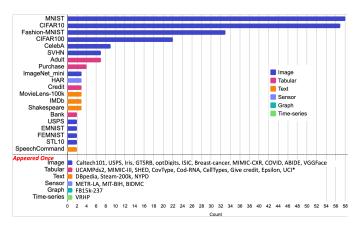


Fig. 4: Dataset choice for experiments.

in recommender systems, opted for graph-based models like NCF and LightGCN. For Bayesian FU implementations, as seen in works like [65, 74], the Bayesian Neural Network (BNN) architecture was used. [94] employed knowledge graph embedding models (TransE, ComplEx, and RotE) specifically tailored for graph-type datasets. A detailed list of model architecture is described in Section A.

To address asynchronous FU, FedHarmony [80] introduced FedBuff for integrating client updates. Previous works had been primarily focused on unlearning in an HFL setting, more recently, from 2023, unlearning was explored in various federated learning settings, such as VFL/cross-silo FL [46, 82–84, 105] and blockchain-based FL [22, 106].

The prevailing aggregation method in FU remains FedAvg, weighing each client's gradient based on their dataset size and averaging them to ensure proportional impact. Some work [2, 56, 58] involved Byzantine-robust aggregation algorithms—Krum [115], Trimmed-Mean [114], and Median [114] to protect the learning system from backdoor or poisoning attacks. These aggregation rules were used to preserve the security or integrity of the model, effectively handling outliers by excluding the most deviating updates [115], trimming a specific percentage of extreme updates, or computing the coordinate-wise median [114].

## E. Research Implication of Prior Works

As summarized in Table IV, researchers primarily consider efficacy, fidelity, and efficiency when designing FU methods. However, due to the distributed nature of FL and the vulnerabilities that arise in FU, they have also sought theoretical guarantees for convergence [19, 36, 71, 97] and performance [28, 29, 37, 41, 42, 50, 73, 81, 103]. Other key concerns include scalability with respect to the number of clients [30, 44, 47, 59, 66, 70], adaptivity to changing data distributions [24, 85, 104], asynchronous parameter update timing [24, 30], compatibility with different model architectures [24, 27, 52, 59, 66], and security against data pollution or leakage, including backdoor and poisoning attacks [2, 28, 29, 31, 56, 58, 75, 76, 81–83, 97]. Recent works [20, 35, 38, 45, 89, 109, 116] have also addressed fairness to prevent unbalanced unlearning [109] and encourage meaningful client participation [38].

<sup>&</sup>lt;sup>5</sup>https://yann.lecun.com/exdb/mnist/

<sup>&</sup>lt;sup>6</sup>https://www.cs.toronto.edu/ kriz/cifar.html

<sup>&</sup>lt;sup>7</sup>https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html

 $<sup>^{8}</sup> https://www.nist.gov/itl/products-and-services/emnist-dataset \\$ 

<sup>&</sup>lt;sup>9</sup>TCGA https://www.cancer.gov/ccg/research/genome-sequencing/tcga TMI https://opencas.webarchiv.kit.edu/?q=tmidataset

TABLE IV: Data Dist. (distribution), NIID sim. (Non-IID simulation), data types, aggregation method, and implications. "Dirichlet," "Fang," "unique," and "random" refer to Dirichlet distribution, Fang [113]'s approach, limited class assignments, and random assignment; "n/d" and "-" denotes not disclosed and none;. **img, tab, txt**, and **oth** refer to the image, tabular, text, and other datasets. **efc, fid, efn, sec, gua, ada**, and **sca** represent efficacy, fidelity, efficiency, security, (theoretical) guarantee, adaptivity, and scalability. 'Med' and 'TrMean' indicate Median, Trimmed mean[114], respectively.

Ref.	Data Dist.	NIID Sim.			Type		Aggregation Method				nplicat			
Exact-Fun [87]	Non-IID	Random	img ●	tab	txt	oth	FedAvg	etc	fid	efn •	sec	gua	ada	sca
FedUHB [50]	n/d Non-IID	Dirichlet	•				FedAvg FedAvg	•	•	•		•		
FATS [88] FedADP [32]	n/d	_			•		FedAvg	•	•		•	•		
Shao et al. [89] Wang et al. [90]	Non-IÍD IID	Unique	•			•	Weighted Avg FedAvg		•	•		•		
FedRecover [2]	Non-IID	Fang	•	•		•	FedAvg, Med, TrMean	•	_	•	•			
Wu et al. [34] Fu et al. [91]	n/d IID, Non-IID	Fang					FedAvg n/d			•				
FedRecovery [29] CFRU [57]	IID Non-IID	n/d	•				FedAvg FedAvg	•	•	•	•	•		
MetaFul [64]	IID, Non-IID	Dirichlet	•		•	•	FedAvg	•		•		•		
Deng et al. [82] Starfish [95]	IID n/d	-		•			n/d FedAvg	•	•		•	•		
Crab [96]	n/d	-	•	_	•		FedAvg	•	•	•		•		
FedEraser [5] FRU [59]	n/d n/d	-	•	•	•		FedAvg FedAvg		•				•	•
SIFU [97] FedUnlearn [99]	IID, Non-IÍD IID	Dirichlet	•				FedAvg FedAvg	•	•	•	•	•		
FedUNRAN [49]	Non-IID	Random					FedAvg		•	•				
FUCRT [69] FedMUA [100]	IID, Non-IID IID, Non-IID	Dirichlet Dirichlet	•				n/d FedAvg	•	•	•	•			
Pan et al. [85]	IID, Non-IID	Random		•			Weighted Avg.	•	•	•		•		
FFMU [103] FedFilter [78]	n/d Non-IID	-	•		•		FedAvg Avg. base layers		•	•		•		
UKRL [31]	IID, Non-IID	Random	•				FedAvg	•	_	•	•			
MoDe [62] ConDa [33]	Non-IID IID, Non-IID	Dirichlet Random					FedAvg FedAvg		•					
RobustFU [35]	IID, Non-IID	n/d	_	•	_	_	n/d	•	•	•	•	•	_	
FRAMU [104] VeriFi [58]	Non-IID Non-IID	Concept drift Dirichlet		•		•	FedAvg FedAvg, Krum, Median		•		•		•	
Lin et al. [106] FC [28]	n/d IID, Non-IID	- - /d	•	_	_		Weighted Avg	•	•	•	•	_	•	
Wang et al. [107]	IID, Non-IID	n/d Fang		•	•		SCMA FedAvg				•	•		
FAST [63] ElBedoui et al. [72]	IID, Non-IID IID	Random	•				FedAvg FedAvg	•	•	•	•			
Fast-FedUL [37]	Non-IID	Dirichlet	•			•	SecAgg	•	•	•		•		
NoT [53] 2F2L [60]	IID, Non-IID IID	Dirichlet	•				FedAvg FedAvg	•	•	•				
Liu et al. [27]	IID	-	•				FedAvg	•	•	•		•	•	
Li et al. [43] BadUnlearn [56]	n/d Non-IID	Dirchlet					FedAvg FedAvg, Med, TrMean				•	•		•
FedU [42]	IID	-	•				n/d	•	•	•	_	•		
FedME2 [71] Alam et al. [75]	n/d IID	-					FedAvg FedAvg		•		•			
BFU [74]	n/d	Covariata abift	•				FedAvg	•	•	•				
FedHarmony [80] Chen et al. [92]	Non-IID IID, Non-IID	Covariate shift n/d				•	FedEqual n/d		•	•				
FCU [40] Ferrari [79]	Non-IID IID, Non-IID	Dirichlet Dirichlet	•				FedAvg n/d	•	•	•				
Imba-ULRc [45]	Non-IID	Random	•			•	Weighted Avg	•	•	•		•	•	
Xu et al. [68] TrustChain [21]	IID, Non-IID Non-IID	n/d n/d	•		•		FedAvg n/d	•	•	•				
FUSED [25]	Non-IID	Dirichlet	•				FedAvg	•	•	•	•			
Goldfish [93] FedLU [94]	Non-IID Non-IID	Random Unique	•			•	n/d FedAvg						•	
FedAF [61]	n/d	· -	•			•	FedAvg	•	•	•	•			
FedQUIT [51] CKGD [55]	Non-IID IID, Non-IID	Dirichlet Dirichlet					n/d FedAvg			•				
SFU [73] HDUS [66]	IID, Non-IID Non-IID	Random	•		•		FedAvg	•	•	•		•	_	
EWC-SGA [19]	IID, Non-IID	Unique Unique					n/d FedAvg		•				•	•
SFU [76] Halimi et al. [101]	IID, Non-IID IID	Dirichlet	•				n/d FedAvg	•	•		•			
QuickDrop [70]	IID, Non-IID	Dirichlet	•				FedAvg	•	•	•				•
LMŘ [67] Wang et al. [77]	Non-IID Non-IID	n/d n/d	•				n/d FedAvg	•	•	•			•	
Han et al. [84]	n/d	· -	•				FedAvg	•	•	•				
FedOSD [41] F2ul [38]	IID, Non-IID IID, Non-IID	Unique n/d	•				FedAvg FedAvg	•	•	•		•	•	
VFU [105]	n/d	· -	•	•			n/d	•	•	•			-	
forgetSVGD [65] CforgetSVGD [26]	Non-IID Non-IID	Unique Unique	•				n/d FedAvg		•	•				
KNOT [30] Lin et al. [108]	Non-IID	Dirichlet	•	•	•		FedAvg, FedBuff	•	•	•		_	•	•
Lin et al. [109]	IID, Non-IID IID, Non-IID	Random n/d	•		•		FedAvg FedAvg	•	•	•		•	•	•
Liu et al. [36] Hier-FUN [44]	n/d Non-IID	Random	•				SecAgg FedAvg	•	•	•	•	•		_
FedUMP [47]	n/d	-	•				Selective Avg.	•	•	•		•		•
k-IPfedAvg [23] FedAU [18]	IID, Non-IID IID, Non-IID	n/d Dirichlet	•				FedAvg, k-IPfedAvg n/d	•	•	•		•		
	1110, 11011-1110	Difficillet					11/ 0							

Notably, while efficacy, fidelity, and efficiency are empirically validated, security, adaptivity, and scalability are often only theoretically guaranteed in the respective papers. Security mechanisms primarily rely on Homomorphic Encryption [82, 83], Differential Privacy [29, 52, 75, 76], or trusted third parties [31] rather than formal security analysis—RevFRF [81] being a notable exception. Although recent blockchain-based approaches [21, 22, 106, 117] aim to enhance scalability and security through their inherent framework design, they still lack strong theoretical guarantees in these aspects.

T-5. Takeaways on Configurations and Implications. Most works aim to achieve efficacy, fidelity, or efficiency, with a few additional focuses on security, guarantee, scalability, and adaptability. For instance, over 90% of them relied on the simplest FedAvg, although it could lead to intolerance to Byzantine failures or heterogeneity. Given increased vulnerabilities due to accessibility to both learned and unlearned models and dynamic client participation, it becomes imperative to consider the other implications actively.

## VI. Existing Unlearning Techniques

Unlearning methods are classified into exact and approximate. Exact unlearning ensures exact indistinguishability of the distributions between unlearned and retrained models, while approximate unlearning guarantees only approximately [6], often due to imprecise estimation of data influence [61].

Despite its precision, exact unlearning comes with substantial overhead compared to approximate one. It necessitates the computation and storage of multiple sub-models during initial training, which can also hinder its adaptability to dynamically changing data [6]. Thus, the goal of exact unlearning is retraining as fast as possible with minimal overhead [50, 87]. Additionally, it is constrained to simpler models due to scalability reasons [118] because it often relies on partitionaggregation frameworks [61].

In contrast, approximate unlearning methods offer greater time efficiency gains; for instance, the computational burden is reduced through calculations on sampled parameters instead of computing over all parameters [27, 60]. Also, model utility can be restored during the performance recovery process. Table V summarizes the unlearning target, influence removal, and performance recovery method for each paper.

## A. Influence Removal

This stage eliminates the unlearning targets' influences from the trained model, such that the unlearned model behaves as if it has never seen the target data. The gray boxes summarize the concepts and limitations.

1) Historical Information: The server stores historical local model updates for potential model restoration or estimation before engaging in target unlearning. In SIFU [97], the server identified and restored an optimal FL iteration without the target information, similar to Starfish [95] and R2S [46]. Estimating the unlearned model includes simply restoring historically saved remaining clients' updates [2, 34, 89] or

applying calibration upon them by approximating the direction of updates to generate an unlearned model by weighted average [90]. Similarly, FedRecovery [29] subtracted a weighted sum of gradient residuals from the global model using historical local updates, weighing them by clients' contribution in decreasing global loss. Meanwhile, some works subtracted target updates from either global [34, 37, 53, 64] or remaining clients' models [54, 82]. Interestingly, Fu et al. [91] also recovered from the historically saved local model updates but projected them to a low-dimensional subspace inspired by that client-specific information is embedded in their higher-dimensional space.

However, storing all updates is often infeasible, especially for those adapting large models or for cross-device FL with millions of clients. Some works, thus, store updates periodically [5] or selectively [57, 59, 96, 98], to relax the memory overhead. FedEraser [5] took a direction calibration approach as in [90] but only using recent sets of updates. Crab [96] and FRU [59] used a rollback mechanism, as in DBMS, to restore the state before the target engagement and retraining with remaining clients, similar to Zuo et al. [21] and CFRU [57]. FRU [59] kept important updates by negative sampling, while Crab [96] only stored high-contributing clients in which the gradients have large KL divergence and high cosine similarity to the global model and identified the least affected gradients to start calibration from. Monitoring the sampling probability, FATS [88] recomputed gradients only when changes occurred after deleting a target to minimize parameter update frequency.

**T-6. Takeaways on Historical Information:** Unlearning using historical information could increase the correctness of the unlearned model, as the unlearned model is likely to converge. Nonetheless, as the model size or the number of epochs gets larger, saving all model updates is infeasible due to its high memory requirement. Thus, sampling methods should be more discovered, finding the parameters that represent and affect the model the most.

2) Data Manipulation: In this approach, noise is added to the target data samples or labels to create a perturbed model [31]. Directly adding noise to the target samples can only be done by the target client, as the raw sample is only accessible to the data-owning client. FedUNRAN [49], for instance, replaced the true labels of the target samples with random ones. FUCRT [69] changed the target samples to a transformation class, redirecting their influence within the model to semantically similar classes. Inspired by neuroscientific principles of memory degradation, Meerza et al. [20] created a confusion set consisting of pairs of a target sample and a wrong label. ConFUSE thereby intentionally confused the model by adding a regularizer term to minimize predictions between the target set and the confusion set, effectively encouraging the model to forget the correct knowledge. Conversely, Imba-ULRc [45] oversampled and generated data that remaining clients owned to weaken the influence of the target samples.

TABLE V: Unlearning targets, influence removal methods, and performance recovery methods define an unlearning mechanism. Each refers to what we want the model to forget, how the model can forget, and how to maintain the performance on the remembered dataset, respectively. **sp, cs, ct**, and **ft** refer to sample, class, client, and feature, respectively.

Ref.	Targe	t	Unlearning M	ethod
ICI.	sp cs c	t ft	Influence removal	Performance recovery
RevFRF[81]		•	Remove the node of the target client and all the child nodes	-
Exact-Fun[87]	•		Retrain the remaining clients using a quantized model	-
FedUHB[50]			Rapid retraining: accelerating using heavy ball method	-
FATS[88]	•		Retrain only if sampling probability changes after removal	-
R2S[46]		•	Retrain from historically saved local model checkpoints	Automatic optimizer control for generalization
Zuo et al.[21]	• • •		Rollback before the client joins, retraining in remaining clients	-
FedADP[32]			Recover from historical local updates based on cosine sim. <sup>10</sup>	<del>-</del>
Shao et al.[89]			Recover from historically saved local updates	Add a penalty term derived from projected gradients
Wang et al.[90]			Recover from historically saved local model updates	
FedRecover[2]			Recover from historically saved remaining clients' updates	Pre- and Post-training
Wu et al.[34]	•		Recover from historically saved remaining clients' updates	KD: transfer remembered knowledge
Fu et al.[91]	9		Recover from historically saved local updates by projection	Post KD iterations on auxiliary data in the server Add Gaussian noise
FedRecovery[29] CFRU[57]		-	Recover from historically saved local model updates Recover from sampled historical updates	Add Gaussian noise
FedADP[46]			Recover from selectively stored historical info	Post-training with remaining clients
MetaFul[64]			Recover from saved updates and subtract target ones	Fine-tuning using the direction of the target updates
Deng et al.[82]		•	Recover from saved target model updates at the last round by	Constraints on intermediate local model parameter
Deng et un[02]			subtracting them from the global model	during training
Starfish[95]			Recover from least-affected local updates	Recompute the local updates with large approx error
Crab[96]	•		Recover from selectively saved least-affected local updates	-
FedEraser[5]			Recover from periodically saved local model updates	Post-training: a few training rounds after unlearning
FRU[59]	•		Restore certain global model updates saved historically	Gradient manipulation: direction
SIFU[97]		-	Restore and perturb certain global models, saved historically	-
FedRemover[98]			Adjust direction from historically saved local updates only if	-
ConFUSE[20]			the model performance drops  Create a fake dataset to confuse the model	Undete calient weights only
Imba-ULRc[45]	•	•	Create a fake dataset to confuse the model Finetuning on oversampled, noise-removed data	Update salient weights only Post-training with remaining clients
FedUnlearn[99]			Perturb target label to the fake one	- ost-training with remaining chems
FedUNRAN[49]			Perturb target label to the random ones	Post-training with remaining clients
FUCRT[69]	• `		Perturb target label to the random ones	Optimize local space to be aligned with global space
FedMUA[100]			Modify target sample prediction	Identify the most influential samples
Pan et al.[85]		•	Perturb target client updates	Gradient alignment
SecForget[102]	•		Perturbation addition to the target client	Add a regularization term: L1 norm
FFMU[103]	•		Perturbation addition to the target client	-
FedFilter[78]	•		Perturbing target update by generating a reverse gradient	SGD to minimize impact on the model accuracy
UKRL[31]			Perturbing target updates by training it on noised input	-
MoDe[62]	• •		Scale down on the target data points	Guided fine-tuning on remaining data points
ConDa[33]			Scale down the global param most impacted by target clients	
RobustFU[35]	•		Scale up/down based on label prediction discrepancies	Introduce random samples based on the discrepancy
FRAMU[104]	•		Scale up/down using an attention mechanism	-
VeriFi[58]		•	Scale up/down on remaining/target clients' updates	-
Lin et al.[106]		•	Scale up/down on remaining clients' updates	-
FC[28] Wang et al.[107]	• •	•	Scale to zero on the target clients' updates Prune the channel of gradients corresponding to the target	Fine-tuning: few FL training without regularization
Xu et al.[68]			Prune less influential channel in identifying target	- The-tuning. Icw 1'L training without regularization
FUSED[25]			Randomly dropout parameters in critical layers	
SecureCut[83]		•	Prune the nodes of their local model and retrain it	Post-training with remaining clients
FAST[63]	Ť	•	Subtract target model updates from the global model	Fine-tuning with a small-size IID dataset
ElBedoui et al.[72]	•		Subtract target model updates from the global model <sup>11</sup>	-
Fast-FedUL[37]			Subtract differences between target model updates and the	_
Tust Tede D[b/]			aggregation of selectively stored local updates	
NoT[53]			Subtract targets' layer-wise parameters from the global model	Post-training with remaining clients
Wang et al.[24]	•	•	Subtract target confidence from a confidence matrix	-
2F2L[60]	•		Loss function approximation	
Liu et al.[27]		•	Loss function approximation using diagonal FIM	Apply Momentum technique to Hessian diagonal
Xie et al.[48]	•		Loss function approximation using FIM	Parameter clipping & KD using synthesized samples
Li et al.[43]			Compute approximate Hessian to estimate recovered gradient	Gradient clipping
BadUnlearn[56]	9	-	Compute approximate Hessian using limited historical data	Post training with remaining = -11t-
Appro-Fun[52]	9		Compute approximate Hessian	Post-training with remaining clients
FedU[42]		•	Approximate influence using Hessian-vector product and subtract it from the global model	local post-training with remaining dataset
FedME2[71]		•	Multi-task learning: data erasure and remembrance	Add a regularization term: L2 norm
Alam et al.[75]	•	•	Multi-task learning: data erasure and remembrance	Dynamic penalization
BFU[74]	•		Multi-task learning: data erasure	Multi-task learning: maintaining performance
FedHarmony[80]		•	Multi-task learning: data erasure and remembrance	- 6 F
Chen et al.[92]	•	_	Multi-task learning: minimize KLD between the predicted	Multiply the loss by a scaling factor to balance
			distribution of forget and arbitrary data	between efficacy and fidelity
Goldfish[93]	•		KD & Multi-task learning: minimize prediction difference	Multi-task learning: maintaining performance
DOT			within the target set and transfer only retaining knowledge	B
FCU[40]			Contrastive unlearning to make it similar to a model trained	Post-training on low-frequency components
Ecdi iIIO41	_		only on the retaining set at feature level	Eine tuning suppressing estimation of the toward date
FedLU[94]	•		KD: transfer knowledge to be remembered	Fine-tuning: suppressing activation of the target data
FedAF[61]	•••		KD: transfer fake knowledge	EWC training, using Hessian matrix as a regularizer
FedQUIT[51]	9		KD: clip guided few shot kd from historically saved updates	KD: transfer retaining knowledge
CKGD [55]	•	•	KD: clip-guided few-shot kd from historically saved updates	KD: transfer retaining knowledge
SFU[73]	•		KD: negative transfer knowledge to be forgotten Ensemble remaining clients' knowledge-distilled seed models	KD: transfer retaining knowledge
HDUS[66]				

EWC-SGA[19] • •	•	EWC training, using FIM as a regularizer
SFU[76] ●	Reverse training: Gradient Ascent (GA)	Project the target updates into the subspace
	<ul> <li>Reverse training: maximizing local loss</li> </ul>	Post-training: a few training rounds after unlearning
QuickDrop[70] •	<ul> <li>Reverse training (SGA) on the distilled dataset</li> </ul>	Fine-tuning on augmented remaining data samples
LMR[67] ●	Reverse training: layer-wise GA	Post training
Wang et al.[77] ●	Reverse training: GA	Post-training
Han et al.[84]	<ul> <li>Reverse training: GA guided by avg'd remaining clients</li> </ul>	Post-training with remaining clients
BlockFUL[22]   •	Reverse training: GA	-
FedOSD[41]	Reverse training: GA to orthogonal steepest descent direction	Project gradient to the original plane
F2ul[38]	Reverse training: GA	post training
forgetSVGD[65] •	Variational inference, maximizing local loss	Post-training with remaining clients
CforgetSVGD[26] ● ●	<ul> <li>Variational inference, quantization and sparsification method</li> </ul>	Post-training with remaining clients
KNOT[30]	Cluster based on their similarity and retrain one cluster	-
Lin et al.[108]	Clustering: retrain the cluster containing the target	-
Lin et al.[109]	<ul> <li>Cluster based on their similarity and calibrate the gradient by step length and direction</li> </ul>	Dynamically select remaining clients possessing unbalanced local data
Liu et al.[36]	• Clustering: retrain low performing cluster without target client	-
Hier-FUN[44]	Clustering and GA on target device	Post training with remaining clients
FedUMP[47]	• Clustering client <sup>12</sup> and remove the subset model	-
k-IPfedAvg[23]	Remove the target clients from the cluster	Retrain when any cluster has $\leq 2/3/4$ # of clients
FedAŬ[18] ● ●	• Add auxiliary unlearning module while training and XOR with	-
	the local model upon unlearning request	
Ferrari[79]	<ul> <li>Guided optimization to minimize feature sensitivity</li> </ul>	-
TrustChain[117]	Finetuning using LoRA	-
10		

To Cosine Similarity. 11 Subtracting model updates calculated only on the target data. 12 based on their distribution

**T-7. Takeaways on Data Manipulation:** Since only the target client has access to the raw data, it can directly manipulate the target samples by adding noise or flipping labels. This confuses the global model toward mispredicting the data to be forgotten. However, this approach raises a fundamental question: does inducing incorrect predictions truly constitute unlearning?

## 3) Gradient Manipulation:

<u>Perturbation:</u> The following approaches directly introduce noise into the model parameters. This can be done both by the target client and the server, as the server also has access to the local model parameters.

In FedMUA [100], the target client modified the predictions for the samples corresponding to the features that need to be forgotten. In [85], the server added adaptively scaled noise to the target client based on feature relevance. FFMU [103] added Gaussian noise to smooth all local models' gradients, treating them as perturbations during server aggregation. In contrast, FedRecovery [29] applied noise to the unlearned model to make retrained and unlearned models indistinguishable. SecForget [102] incorporated a trainable dummy gradient generator for each client, simulating the neurons of a model to eliminate memory of specific data. Meanwhile, in FedFilter [78], the server generated a random reverse gradient and performed SGD to maximize the elimination effect.

<u>Scaling:</u> In Verifi [58], target clients' gradients were downscaled, and remaining clients' gradients were upscaled, causing gradual vanishment of the target's influence. MoDe [62] constructed a shadow model of the global model, sent exclusively to and trained on remaining clients. The unlearning model reduced discrimination towards target data points by adding  $(1 - \lambda)$  scaled weights of the shadow model to the  $\lambda$ , within the range [0, 1], scaled unlearning model. FRAMU [104] involved clients in calculating and sending attention scores with local model updates. These scores were utilized on the server to assign less weight to model updates corresponding to the target data. Similarly, Lin et al. [106] adaptively assessed the remaining clients' contributions while training the

model, and the server performed a weighted average on the remaining clients' updates based on the contribution score.

<u>Pruning:</u> In SecureCut [83], target clients pruned the nodes of their tree-structured local models before retraining them. ElBedoui et al. [72] computed model parameters from target data and sent them to the server, which is subtracted from the global model. Similar to [68] that prune less influential channels in identifying the target class, Wang et al. [107] calculated relevant scores between channels and categories using TF-IDF and pruned the most discriminative channels of the target category. Similarly, FUSED [25] identified the most sensitive layers through layer-wise analysis and applied an unlearning adapter that randomly dropped out some parameters within the selected layer. This approach improved efficiency by limiting unlearning and aggregation to selective layers.

**T-8.** Takeaways on Gradient Manipulation: Models are perturbed such that they fail to achieve the task for the target information. Local updates are scaled differently to minimize and maximize the importance of the target model and retaining knowledge, respectively. Pruning refines the gradients or nodes containing the target knowledge. Nonetheless, all gradient manipulation methods incur additional complexity in finding precise perturbation, scaling factors, and pruning information because the model will not converge otherwise.

4) Loss and Influence Approximation: Loss functions are approximated by calculating the inverse Hessian matrix as if the unlearned model had not been trained on the target data. The inverse Hessian, a square weighting matrix, scales gradients based on second-order partial derivatives between the training set mean point and the sample of interest [119]. It describes the local curvature of the loss function over the entire training dataset such that the optimizer takes a more aggressive step in shallow one [120]. Nonetheless, directly calculating the inverse Hessian incurs substantial memory and computational complexity, making it infeasible in practice [121].

To address this, researchers have explored efficient approx-

imation methods. Liu et al. [27], for instance, used the Quasi-Newton method, computing the Hessian matrix with the diagonal empirical Fisher Information Matrix (FIM) to estimate recovered gradients, similar to BadUnlearn [56] and Li et al. [43]. Appro-Fun [52] further incorporated differential privacy to mitigate data leakage. Beyond FIM-based approximations, Xie et al. [48] adaptively clipped global model parameters strongly related to the target data. Meanwhile, 2F2L [60] relied on Taylor expansion, optimizing quadratic loss functions using neural tangent kernels and a Newton step to compute optimal weights based on the remaining dataset.

Even FedU [42] proposed a method that estimates the influence of target data points using gradients and Hessian-Vector Products, requiring only the target samples. This method is more efficient and cost-effective since it does not depend on access to the entire remaining training dataset.

**T-9.** Takeaways on Loss and Influence Approximation: Although an inverse Hessian allows scaling the gradients by approximating the loss function, naively calculating it incurs an expensive computational burden. Researchers thus approximate the inverse Hessian to reduce the computational complexity, although there remains more room for improvement to find a more efficient and more accurate approximation.

5) Knowledge Distillation (KD): In contrast to Goldfish [93] and CKGD [55], which used knowledge distillation (KD) at the server to transfer only retained knowledge to the unlearned (student) model, FedAF [61] and VFU [105] adopted a teacher-student learning pattern at the target client. In this approach, a teacher model generated fake labels, and the student model learned the manipulated knowledge. Eventually, the teacher model—never exposed to the original feature-label pair—became the unlearned model, avoiding data leakage.

FedLU [94] introduced a loss function based on retroactive inference theory, employing mutual knowledge distillation to for unlearning while handling covariate drift between local optimization and global convergence during training. FedQUIT [51] constructed a virtual teacher to approximate the global model's knowledge without influence from the target clients, while SFU [73] incorporated three teacher models: one for forgetting via negative transfer to ensure random predictions of the student model on target data and two for performance preservation.

In a fully decentralized FU framework, HDUS [66], all clients collaboratively generated a shared reference dataset that contained unlabeled and non-client-specific data. The remaining clients then trained their seed models on this dataset and achieved unlearning through a model ensembling. The use of seed models addressed concerns in the federated context by reducing the communication burden, minimizing data leakage risks, and allowing local models to have distinct architectures.

**T-10. Takeaways on knowledge distillation:** Knowledge distillation reduces the training dataset or model size so that it achieves better efficiency than retraining.

It is especially beneficial in that the teacher model does not have the original pair of knowledge; there would be the least residual information about the target data, providing security along with efficacy.

6) Multi-task Learning: Multi-task learning balances retaining essential knowledge while discarding target knowledge in three ways. First, the unlearned models aim to maximize indistinguishability within a forget set as in Goldfish [93] or between it and an arbitrary dataset [92]. For another line of work, BFU [74] leveraged hard-parameter sharing and variational Bayesian unlearning to optimize an approximate posterior of the remaining dataset. Similarly, Alam et al. [75] focused on eliminating compromised data, augmenting loss separately for benign and spoiled data to preserve non-malicious behavior while unlearning trigger patterns. Finally, FedME2 [71] optimized the local model to ensure both classification accuracy and a memory evaluation loss, derived from a lightweight evaluation model within each client, detecting whether data to be forgotten is remembered. FedHarmony [80] was trained for three tasks: to update the label predictor, to discriminate between sites, and to remove the site-specific knowledge by penalizing deviation in the probability of the outputs from a uniform distribution. FCU [40] employed contrastive learning; specifically, they adjusted the model parameters to minimize the difference between the current global model (negative examples) and a reference model (positive example) that has never seen the target data.

**T-11. Takeaways on Multi-task Learning:** The tasks typically involve target removal and performance restoration. It can be achieved sequentially or in parallel, where the latter achieves better time efficiency.

7) Reverse Training: Gradient ascent, instead of descent, is employed either stochastically by the server on the global model [19, 38] or by the target client on the local model [22, 44, 67, 76, 77, 84, 101, 105] to reverse the original training process. This approach is different from the subtraction of historically stored parameters in that it trains a target model to achieve maximal loss through ascent.

Rather than simply flipping the gradient sign, Halimi et al. [101] utilized projected gradient ascent to maximize local empirical loss. Similarly, Han et al. [84] computed ascent directions guided by the averaged gradients of the remaining clients. Building on this idea, FedOSD [41] applied gradient ascent along the orthogonal steepest descent direction to prevent conflicts with remaining clients' gradients, enhancing unlearning while preserving model utility.

In QuickDrop [70], each client generated a distilled dataset—a small synthetic dataset that condenses critical information from the learning phase. They subsequently fine-tuned the gradients computed from this distilled data to align with their original local datasets.

**T-12. Takeaways on Reverse Training:** Reverse training, represented by (projected/stochastic) gradient ascent, gives the most intuition for unlearning by optimizing the loss to be maximal. As it harms the original performance by a large margin, it is often accompanied by performance recovery.

8) Clustering: The server clustered clients to maintain the minimal number of clients to perform retraining, limited to the only cluster to which the requesting client belonged as in Liu et al. [36]. Lin et al. [108], for instance, clustered the clients into multiple shards prior to training and adopted coded computation to reduce the storage overhead by compressing the model parameters across distinct shards. FedUMP [47] partitioned clients based on their label distribution and unlearning cost and retrained only a subset of the partitioned clients. Similarly, in [109], Hier-Fun [44], and KNOT [30] grouped clients based on their similarities in model disparity (angle) and short training time to reduce the number of computations.

**T-13. Takeaways on Clustering:** This approach minimizes the number of clients requiring retraining, making it more time-efficient than involving all clients. However, finding an optimal clustering is challenging and may become infeasible with frequent unlearning requests from random clients.

9) Others: Ferrari [79] minimized target feature sensitivity by quantifying the model's response to input changes using a Lipschitz continuity metric. Wang et al. [24] proposed an unlearning method in VFL, where clients and the server maintain confidence matrices. Target clients compute differences in confidence vectors before and after unlearning, and the server updates its matrix and gradients accordingly. FedAU [18] introduced an auxiliary unlearning module from the beginning of the training process, XORing its parameters with the target model to eliminate its influence. TrustChain [21] is the first, to our knowledge, to explore federated LLM unlearning. It employs blockchain to track data contributions and uses LoRA hyperparameter adjustments for targeted unlearning while ensuring transparency and accountability.

## B. Performance Recovery

Removing influence from a trained model could deteriorate its performance on the remaining dataset because target erasure often involves undoing specific parameters [6], thereby leaving other parameters incomplete [107]. These performance drops become more noticeable when the model parameters are manipulated to remove influence. As such, additional approaches are explored to restore the performance of the unlearned model on the remaining dataset.

1) Post Training: Some approaches [2, 24, 26, 38, 45, 49, 53, 65, 67, 83, 84, 101] applied post-training to the unlearned model after influence removal. Wang et al. [77] referred to this process as boosting training, where the remaining clients and the server perform a few additional global rounds of standard

FL training—without the target influence—to restore model performance. FedU [42] conducted post-training simultaneously with influence removal by training the unlearned local model on the remaining local dataset. Unlike most clustering-based unlearning techniques, Hier-Fun [44] performed a few steps of post-training with the remaining clients to restore performance, even in highly heterogeneous edge environments. Notably, post-training does not involve additional data points.

2) Fine-tuning: Building on the neurological theory that memory traces fade, some works have focused on reducing the discriminability of target data while maintaining that of the remaining data. For instance, some approaches removed the regularization term [107] or suppressed the activation of forgotten knowledge [94].

Zhao et al. [62] employed guided fine-tuning, similar to MetaFul [64], where the server additionally sent a degradation model to the target client. This model generated pseudolabels, assisting the client in restoring its discriminability on the remaining data. FUCRT [69] fine-tuned on a newly created dataset consisting of class-transformed target data. Specifically, the authors reassigned labels to the second most probable class, preserving the original global model's high performance with minimal loss. During fine-tuning, they also incorporated contrastive loss to maintain feature separability. Similarly, QuickDrop [70] and FAST [63] used post-training on small datasets to recover performance. FCU [40] applied fine-tuning selectively, retaining low-frequency model components, which encode general knowledge, and adjusting highfrequency components tied to specific client data. This ensured the model could forget the targeted data without compromising overall performance.

3) Gradient Manipulation: Some achieved performance recovery by directly manipulating the magnitude or direction of gradients. Li et al. [43] clipped the recovered gradients to limit errors that could degrade model performance and to preserve generalization, similar to the adaptive clipping method used by Xie et al. [48]. In FRU [59], the remaining clients run additional local training and send the new model updates to the server. Then, the server constructed calibrated updates by combining the original local updates with the direction of the new updates. ConFUSE [20] selectively updated only salient weights during unlearning to minimize the impact on the model's performance on retained data, achieving both performance preservation and computational efficiency.

Rather than using redirection, some methods projected the target model's gradients into the orthogonal subspace of the input space [41, 76]. If the unlearning gradients directly conflict with useful learning updates (i.e., pointing in the opposite direction), applying them without modification can disrupt the model's overall accuracy. Projection ensures that unlearning does not interfere with the useful updates from the remaining data, minimizing performance degradation. However, precise gradient manipulation is crucial for maintaining model convergence.

4) Regularization: Regularization or penalty terms, in the forms of Scaling factor [27], L1 norm [102], or L2 norm [71], were applied to the unlearning model to prevent overfitting to unlearning tasks. Elastic Weight Consolidation (EWC) [82],

FIM [19], and the Hessian matrix [61] were also often used as the regularization term to limit the magnitude of parameter updates, ensuring minimal changes to important parameters. Instead of a static penalty term, dynamic penalization approaches, including leveraging a momentum technique [27], were also explored [75]. Similar to [82] that used clients' intermediate parameters computed using EWC, [89] used a penalty term derived from projecting an intermediate-term on the tangent space of the aggregated updates.

5) Knowledge Distillation: Similar to influence removal, fidelity is preserved by transferring only the knowledge to be remembered to the unlearned (student) model [34, 51]. Wu et al. [34] used the original global model as a teacher to train a skewed student model, aiming to enhance both generalization and security. Additionally, the server employed unlabeled data to correct skew introduced during the influence removal process, as seen in [91], by conducting additional knowledge distillation iterations on auxiliary data at the server.

SFU [73] leveraged two teacher models for performance preservation, similar to the aforementioned works, while also incorporating label-based preservation to prevent over-unlearning. The second teacher model distilled label-aware knowledge to minimize unintended information loss. Specifically, it transferred information about structural similarities in the data distribution, allowing the model to differentiate between data that should be forgotten and similar data (e.g., sharing the same label) that should be retained. As a result, the student model preserved essential decision boundaries, reducing the risk of accidental over-unlearning.

In addition to gradient clipping, Xie et al. [48] employed data-free knowledge distillation to restore deteriorated performance. They synthesized pseudo-samples to replace real data for distillation. A teacher model, trained prior to unlearning, then guided the unlearned model using soft labels on these pseudo-samples, helping recover generalization ability. This data-free knowledge distillation enables the server to guide fine-tuning without requiring access to any raw client data.

**T-14.** Takeaways on Performance Recovery: Performance recovery restores the model performance tampered with while forgetting. It is conducted sequentially or in parallel to influence removal: post-training, fine-tuning, and gradient manipulation are followed by the influence removal, while regularization using momentum or an additional term is applied during unlearning and thereafter. Note that knowledge distillation improves not only performance by generalization but also security.

## VII. COMPARISON OF EVALUATION METRICS

Unlearning methods are typically evaluated across three dimensions: efficacy (the effectiveness of unlearning/forgetting target information), fidelity (the accuracy in maintaining performance on remaining data), and efficiency (the comparison to retraining from scratch). A variety of evaluation metrics are summarized in Table VI. For brevity, we have limited the works cited here with a full reference list for each metric provided in Section A.

## A. Efficacy

The efficacy is defined by how accurately or effectively the unlearned model forgot the target information.

1) Performance Metrics: Accuracy [5, 19, 29, 31, 35, 38, 44, 49, 55, 58, 62, 63, 65, 68, 70, 74, 79, 85, 97, 100, 101, 107], F1 score [69], loss [5, 58], and error [40, 103] on the forget set are commonly used to measure the efficacy of influence removal. Statistical metrics such as Mean Squared Error (MSE) and Mean Absolute Error (MAE) have also been employed [81, 104].

For recommendation tasks, in addition to Normalized Discounted Cumulative Gain at rank 10 (NDCG@10), which was used in [90], Yuan et al. [59] and CFRU [57] also utilized Hit Ratio at rank 10 (HR@10) to evaluate the top-10 recommendation performance.

2) Parameter Differences: Kullback–Leibler divergence [58, 74], L2 distance [74, 104], first Wasserstein distance [83], and cosine similarity [37, 57, 99] between the unlearned and retrained models serve as indicators of unlearning efficacy.

FedEraser [5] measured parameter deviation  $\theta$  with respect to the retrained model using:  $\arccos\frac{w_uw_r}{||w_u||||w_r||}$  where  $w_u$  and  $w_r$  are the last-layer weight of the unlearned and retrained model, respectively. Symmetric Absolute Percentage Error (SAPE) [27], defined as  $\frac{|Acc^u_{lest}-Acc^*_{test}|}{|Acc^u_{est}|+|Acc^*_{est}|}$  was also used, where  $Acc^u_{test}$  and  $Acc^*_{test}$  denote test accuracy of retrained and unlearned model, respectively [46, 52]. Gong et al. [26] used Expected Calibration Error (ECE) to assess model calibration by quantifying the difference between actual test accuracy and confidence levels. Similarly, Elbedoui et al. [72] measured the L2 norm between confidence distributions. Meanwhile, Shao et al. [89] designed a custom metric based on the performance difference between the unlearned and retrained models.

3) Indiscrimination Quality: The most commonly used metric in this category is attack-based evaluation, which assesses how effectively unlearned models mitigate Backdoor Attacks (BAs) and Membership Inference Attacks (MIAs).

In BAs, a backdoor trigger is injected into a portion of the training data, and the samples are relabeled to a predetermined class, designating them as target samples. An FL model is then trained to perform two tasks: (1) a backdoor task (mispredicting compromised samples) and (2) a main task (maintaining normal behavior on all other samples). The goal of an unlearning model is to forget backdoor-triggered data and the associated information embedded in the model. Instead of using simple BAs, Alam et al. [75] employed stealthier and longer-lasting backdoors, such as Constraint-and-Scale [122] and Neurotoxin [123]. MIAs, on the other hand, test the unlearning model's ability by identifying whether a target sample was part of the original training dataset.

Attack Success Rate (ASR) is a commonly used metric calculated as the fraction of successful attacks over total attempts. ASR represents the proportion of misclassified backdoor-triggered samples in BAs [2, 57, 91, 100, 101] and incorrect membership predictions in MIAs [52, 70, 79, 85, 101]. Other evaluation metrics include precision [48, 88, 98], recall [5, 48, 54, 98], and F1-score [108, 109]. Xu et al. [68] and Ferrari [79] performed model inversion attacks to assess whether the forgotten target could still be reconstructed from the model

TABLE VI: A summary of Evaluation Metrics

Objectiv	e Category	Metric
Efficacy	Performance	Accuracy, F1, or AUC on the target set Loss and errors on the target set MSE and MAE
	Parameter difference	L2 distance KLD Error rate (SAPE, ECE) Angular deviation 1st Wasserstein distance
	Indiscrimination quality	on ASR, precision, and recall on BA ASR, precision, and recall on MIA Model inversion attack Multi-task learning Influence function Prediction overlap
Fidelity	Performance	Accuracy, F1, or AUC on test set Loss or errors on test set Accuracy, F1, or AUC on remaining set Loss or errors on remaining set
Efficiency	y Complexity	Time taken for unlearning Speed-up ratio Memory in MB

parameters or gradients. Successful unlearning should drive these values toward zero.

Some works compared prediction overlap [51] or attention maps [47, 55, 79] before and after unlearning. For instance, Ferrari [79] used attention maps to determine whether the unlearned model still focuses on the forgotten features. Similarly, CKGD [55] leveraged Grad-CAM to evaluate how the model's primary attention shifts after unlearning.

Meanwhile, FedHarmony [80] aimed to minimize domain classification accuracy while maximizing prediction accuracy. Verifi [58] used influence functions to measure the residual impact of target samples on the unlearned model. Zhu et al. [94] evaluated unlearning effectiveness in entity link prediction tasks by measuring Hits@N and Mean Reciprocal Rank (MRR)—where lower values on the forgetting set indicate better unlearning efficacy.

# B. Fidelity

Maintaining performance is as important as precise unlearning because otherwise, the unlearned model becomes useless. **Performance metrics** (i.e., accuracy, loss, and error rates) on the remaining dataset or test dataset were used to evaluate the model's fidelity. Specifically, the former is to show no drop in accuracy [29, 38, 70, 79, 85], or no rise in errors [21, 77, 85] on the remaining dataset, and the latter is to ensure its overall performance, including generalization capability [23, 35, 38, 79, 97]. The higher accuracy or lower loss indicates better fidelity.

## C. Efficiency

Computational or communications overheads are measured by comparing the time or memory taken for unlearning and retraining. **Time taken** were measured in the number of rounds [35, 41, 85, 97], in seconds [23, 52, 54, 79, 117], in minutes [66], or in CPU time [63]. A speed-up ratio is also used;

similar to [62], [2, 46] calculated the average cost-saving percentage calculated by  $\frac{T-T_r}{T} \times 100$ , where T denotes the number of rounds to retrain a model, and  $T_r$  refers to that taken to train the unlearned model. **Memory** in byte [36, 37, 58, 68] was measured to see if the unlearning method did not incur memory overhead during communication.

**T-15. Takeaways on Evaluation Metrics.** Evaluations rely on indirect measurements, comparing performance, parameters, indiscrimination quality, and complexity between unlearned and retrained models. The absence of a benchmark or unified metric for "unlearnability" hampers fair comparisons across different unlearning methods using diverse architecture or aggregation methods.

**T-16. Takeaways on Attacks Used.** Evaluations using simple BAs are limited due to the vanishing attacks over training rounds and Byzantine-robust aggregation removing those influences. Thus, relying solely on simple BAs may obscure the impact of unlearning, making it challenging to determine its effectiveness or whether the attack has naturally diminished.

## VIII. GAPS AND FUTURE DIRECTIONS

Observing Figure 5, we can see that the research focus in this area is getting diverse and deeper. Followed by the first work [102] in 2020, FU techniques have evolved from simple ML models, including Random Forest and shallow CNN. As time passes, in addition to demonstrating the effectiveness of the technique in non-IID settings, which is a common assumption in FL, more data types and, thus, more model architectures are being used, such as GPT or ViT. This year, authors are using more practical datasets, along with considering a wider range of research implications, such as fairness. This section summarizes research gaps and challenges that have not sufficiently been explored and potential direction. Lack of realistic non-IID data (T-1, T-2, T-3). A model successfully unlearned in an IID setting often fails to properly forget or even converge when applied to practical non-IID data, compromising both efficacy and fidelity. As in standard Federated Learning, information in FU is distributed across multiple clients, and neither the server nor other clients have access to raw data or an understanding of the distributional differences among clients. While recent works increasingly consider heterogeneous settings, many still rely on simulated non-IID distributions generated from originally IID datasets. A commonly used simulation approach involves the Dirichlet distribution, which controls the degree of non-IIDness through a concentration parameter  $\alpha$  [124]. This method is widely adopted due to its ability to approximate real-world data heterogeneity. However, the choice of  $\alpha$  is often arbitrary, with values ranging from 0.1 to 5 (Table IX), and an optimal value that accurately reflects real-world non-IID remains unclear.

To bridge this gap, future work could explore more realistic non-IID scenarios. This includes identifying appropriate  $\alpha$  values, using real-world federated datasets such as FEMNIST,

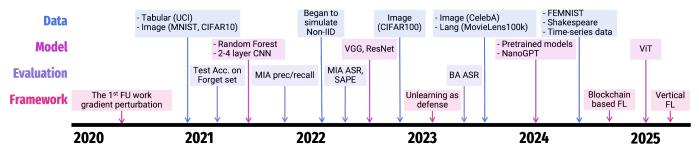


Fig. 5: Emergence of the Research Focus over Time.

Shakespeare, or StackOverflow, or constructing new benchmarks that more accurately capture the complexity of practical non-IID distributions.

Limited application of FU methods beyond vision tasks (T-1, T-3, T-4). As shown in Figure 4 and Table IV, current research on FU predominantly focuses on vision tasks—particularly image classification—while overlooking its broader applicability. This narrow focus may stem from the limited exploration of FL with LLMs, as few studies have evaluated FU methods on language tasks or datasets—even with the rise of powerful models like GPT-4. Moreover, only one study to date has utilized a multivariate dataset, highlighting a significant research gap.

Expanding FU research beyond vision tasks is both necessary and promising. Incorporating diverse tasks from other domains—such as next-word prediction for personal devices (text) or privacy-sensitive medical analysis (text, image, or multivariate data)—could showcase the true potential of FU in preserving privacy and improving efficiency over retraining.

Difficulties in unlearning the closed-source LLMs (T-1, T-4). Proprietary LLMs have undergone training processes that require substantial computational resources and vast amounts of data, both public and private. To protect intellectual property and sensitive information, companies often withhold access to both the training data and model parameters—despite facing lawsuits over the use of copyrighted content in training. As data continues to evolve and removal requests grow, relying solely on post hoc fine-tuning becomes less effective due to memorization issues. Full retraining, meanwhile, is largely infeasible given the scale of modern LLMs and their datasets.

The development trajectory of LLMs is expected to shift toward ensembling multiple pre-trained and/or new models, leveraging their powerful capabilities without exposing privacy-sensitive training data. In this context, a small, well-trained model on copyrighted data could be treated as a target client, with the original LLM acting as the global model. FU could then be employed to remove private data from the original LLM, providing a more efficient alternative to retraining and a more effective solution than fine-tuning.

Underexplored attacks and defense methods in FU (T-2, T-5. The vulnerability of the FU framework has not yet been fully explored or defended against. Given its distributed nature, FU inherits the same risks as FL—such as poisoning and backdoor attacks—but attack and defense strategies remain under-investigated. To date, only two works [35, 56] have proposed a robust FU method addressing poisoning attacks in a binary classification task. More critically, FU may para-

doxically compromise privacy despite its goal of enhancing it. Since learned and unlearned models are shared among participants, adversaries can exploit the differences to infer forgotten information through inference or model inversion attacks.

Example Attack Scenario: Assume the remaining clients and the server are honest-but-curious. In this case, only the target client can verify if unlearning was successful. Other clients could send random or crafted parameters, which the server aggregates—potentially helping a malicious client infer the forgotten data. Worse, if the server itself is malicious, it can manipulate updates and extract sensitive data from both the target and remaining clients.

This vulnerability highlights the importance of minimizing the exact information revealed during parameter sharing to protect against inference attacks. However, no existing FU methods have been rigorously analyzed for such risks. While some works have introduced Homomorphic Encryption or differential privacy for obfuscation, establishing a trusted third party or secure execution environment could also be a viable defense strategy.

Limited adoption of advanced FL techniques (T-3, T-5). Section 4 outlined the unique challenges of unlearning in the federated context. While some of these challenges—such as data heterogeneity and Byzantine failures—are actively being addressed in FL through advanced aggregation methods, many remain unexplored in FU. Notably, most existing FU works still rely on the simplistic FedAvg, which averages all clients' gradients. Only a few studies—FedHarmony [80], FedLU [94], and KNOT [30]—have adopted alternatives like FedProx, FedBuff, and FedEqual. Similarly, FedRecover [2] and VeriFi [58] integrated Byzantine-robust aggregation algorithms, such as Krum and trimmed-mean.

Despite these examples, only a limited number of FU methods have leveraged FL's existing solutions, and the impact of different aggregation strategies on unlearning performance remains underexplored. It would be prudent to first assess whether established FL techniques can help mitigate FU-specific challenges before developing more complex or specialized solutions.

Lacking consideration on scalability and fairness challenges in FU (T-5). In the federated context, where the number of participating clients can range from a hundred (cross-silo) to millions (cross-device), communication and computation bottlenecks can significantly impact system performance. Scalability is thus a critical consideration. Another challenge is fairness and adaptability, as clients frequently join and leave

the system due to reasons like unstable network connections or attempts to exploit system instability. Even in such cases, relevant models should fully unlearn the target, requiring an approach capable of continuous and dynamic unlearning as clients and data evolve.

To mitigate malicious dropout attempts, fairness research in FL has focused on incentivizing/penalizing clients to promote voluntary participation. Similarly, FU methods need to be scalable, fair, and adaptable, ensuring effective unlearning as the client set and data evolve. This involves developing dynamic unlearning techniques that can efficiently forget specific data or clients without the need for retraining from scratch.

Absence of a common benchmark (T-15). Currently, there is no standardized approach for comparing FU methods, which limits comprehensive assessment of their efficacy, fidelity, and efficiency. Most evaluations rely on indirect comparisons between retraining and unlearning performance. However, differences in model architectures, aggregation methods, and hyperparameters across studies introduce inconsistencies that hinder fair comparisons. As shown in Table IV, such variations significantly influence model performance, making it difficult to reach consensus on which methods perform best.

FU research lacks benchmark datasets and unified evaluation criteria. Establishing consistent metrics and experimental setups would enable fair comparisons, helping identify the most effective methods and accelerating progress in the field. Unreliable assessment using naive backdoor attacks (T-16). Existing evaluation methods using backdoor attacks (BAs) have primarily relied on simple attacks, whose effects naturally diminish over successive training rounds. This makes it difficult to determine whether the observed mitigation is due to effective unlearning or the natural weakening of the attack. While some attacks are designed to bypass Byzantinerobust aggregation by identifying and excluding malicious gradients, only one study [75] evaluated unlearning efficacy using more sophisticated BAs such as constraint-and-scale [122] and Neurotoxin [123].

Incorporating advanced BAs is crucial for accurately demonstrating the effectiveness of unlearning, ruling out alternative explanations like natural attack decay. Future FU research should prioritize evaluating methods against a wider range of advanced BAs to provide a more rigorous and comprehensive assessment of robustness.

Underexplored FU methods and settings Most existing studies assume the HFL setting, where local model parameters are exchanged. In contrast, VFL, which involves exchanging intermediate outputs between parties with feature-level partitioned data, remains largely unexplored in the context of unlearning. VFL is particularly relevant for building collaborative models among entities with aligned interests, such as banks, insurance companies, and e-commerce platforms. Despite its potential, only a few works have examined unlearning in VFL or fully decentralized FL settings.

Future research could investigate unlearning techniques applicable to different types of FL, including VFL and fully decentralized architectures. Broadening the scope to assess the efficacy and feasibility of unlearning across diverse FL scenarios could open up new possibilities for collaborative

analytics without compromising data privacy. This direction holds promise for advancing FU and addressing real-world challenges in cross-organizational data collaboration.

## IX. Conclusion

While the right to be forgotten inspired MU and FU, systematic reviews of FU methods are currently limited in their number and depth. In this paper, we give special attention to the "federated" context, highlighting unique complexities, and explore the existing literature in multiple dimensions, including the entity initiating unlearning, methodologies, and limitations. We also highlight challenges in heterogeneity simulation, implications, and dataset usage statistics. This paper also provides valuable insights and directions for future research in the intricate landscape of FU.

#### REFERENCES

- [1] 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.
- [2] 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, 2023, pp. 1366– 1383.
- [3] T. T. Nguyen, T. T. Huynh, P. L. Nguyen, A. W.-C. Liew, H. Yin, and Q. V. H. Nguyen, "A survey of machine unlearning," arXiv preprint arXiv:2209.02299, 2022.
- [4] 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.
- [5] G. Liu, X. Ma, Y. Yang, C. Wang, and J. Liu, "Federaser: Enabling efficient client-level data removal from federated learning models," in 2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS). IEEE, 2021, pp. 1–10.
- [6] H. Xu, T. Zhu, L. Zhang, W. Zhou, and P. S. Yu, "Machine unlearning: A survey," *ACM Computing Surveys*, vol. 56, no. 1, pp. 1–36, 2023.
- [7] L. Wu, S. Guo, J. Wang, Z. Hong, J. Zhang, and J. Zhou, "On knowledge editing in federated learning: Perspectives, challenges, and future directions," *arXiv* preprint arXiv:2306.01431, 2023.
- [8] F. Wang, B. Li, and B. Li, "Federated unlearning and its privacy threats," *IEEE Network*, 2023.
- [9] J. Yang and Y. Zhao, "A survey of federated unlearning: A taxonomy, challenges and future directions," *arXiv* preprint arXiv:2310.19218, 2023.
- [10] Z. Liu, Y. Jiang, J. Shen, M. Peng, K.-Y. Lam, X. Yuan, and X. Liu, "A survey on federated unlearning: Challenges, methods, and future directions," *ACM Computing Surveys*, 2023.
- [11] T. Shaik, X. Tao, H. Xie, L. Li, X. Zhu, and Q. Li, "Exploring the landscape of machine unlearning: A

- comprehensive survey and taxonomy," *IEEE Transactions on Neural Networks and Learning Systems*, 2024.
- [12] N. Romandini, A. Mora, C. Mazzocca, R. Montanari, and P. Bellavista, "Federated unlearning: A survey on methods, design guidelines, and evaluation metrics," *IEEE Transactions on Neural Networks and Learning* Systems, 2024.
- [13] J. Nguyen, K. Malik, H. Zhan, A. Yousefpour, M. Rabbat, M. Malek, and D. Huba, "Federated learning with buffered asynchronous aggregation," in *International conference on artificial intelligence and statistics*. PMLR, 2022, pp. 3581–3607.
- [14] 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.
- [15] L. Graves, V. Nagisetty, and V. Ganesh, "Amnesiac machine learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 13, 2021, pp. 11516–11524.
- [16] H. Huang, X. Ma, S. M. Erfani, J. Bailey, and Y. Wang, "Unlearnable examples: Making personal data unexploitable," in *International Conference on Learning Representations*, 2021.
- [17] A. Peste, D. Alistarh, and C. H. Lampert, "SSSE: Efficiently erasing samples from trained machine learning models," in *NeurIPS 2021 Workshop Privacy in Machine Learning*, 2021.
- [18] H. Gu, G. Zhu, J. Zhang, X. Zhao, Y. Han, L. Fan, and Q. Yang, "Unlearning during learning: An efficient federated machine unlearning method," *arXiv preprint arXiv:2405.15474*, 2024.
- [19] L. Wu, S. Guo, J. Wang, Z. Hong, J. Zhang, and Y. Ding, "Federated unlearning: Guarantee the right of clients to forget," *IEEE Network*, vol. 36, no. 5, pp. 129–135, 2022.
- [20] S. I. A. Meerza, A. Sadovnik, and J. Liu, "Confuse: Confusion-based federated unlearning with salience exploration," in 2024 IEEE Computer Society Annual Symposium on VLSI (ISVLSI). IEEE, 2024, pp. 427– 432.
- [21] X. Zuo, M. Wang, T. Zhu, L. Zhang, D. Ye, S. Yu, and W. Zhou, "Federated trustchain: Blockchainenhanced llm training and unlearning," *arXiv preprint arXiv:2406.04076*, 2024.
- [22] X. Liu, M. Li, X. Wang, G. Yu, W. Ni, L. Li, H. Peng, and R. Liu, "Blockful: Enabling unlearning in blockchained federated learning," *arXiv preprint arXiv*:2402.16294, 2024.
- [23] A. K. Varshney and V. Torra, "Efficient federated unlearning under plausible deniability," *Machine Learning*, vol. 114, no. 1, p. 25, 2025.
- [24] L. Wang and L. Wang, "Forgetting any data at any time: A theoretically certified unlearning framework for vertical federated learning," *arXiv* preprint *arXiv*:2502.17081, 2025.
- [25] Z. Zhong, W. Bao, J. Wang, S. Zhang, J. Zhou, L. Lyu,

- and W. Y. B. Lim, "Unlearning through knowledge overwriting: Reversible federated unlearning via selective sparse adapter," *arXiv preprint arXiv:2502.20709*, 2025.
- [26] J. Gong, O. Simeone, and J. Kang, "Compressed particle-based federated bayesian learning and unlearning," *IEEE Communications Letters*, vol. 27, no. 2, pp. 556–560, 2022.
- [27] 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.
- [28] C. Pan, J. Sima, S. Prakash, V. Rana, and O. Milenkovic, "Machine unlearning of federated clusters," *arXiv* preprint arXiv:2210.16424, 2022.
- [29] L. Zhang, T. Zhu, H. Zhang, P. Xiong, and W. Zhou, "Fedrecovery: Differentially private machine unlearning for federated learning frameworks," *IEEE Transactions* on *Information Forensics and Security*, 2023.
- [30] N. Su and B. Li, "Asynchronous federated unlearning," in *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*. IEEE, 2023, pp. 1–10.
- [31] R.-Z. Xu, S.-Y. Hong, P.-W. Chi, and M.-H. Wang, "A revocation key-based approach towards efficient federated unlearning," in 2023 18th Asia Joint Conference on Information Security (AsiaJCIS). IEEE, 2023, pp. 17–24.
- [32] Y. Jiang, X. Tong, Z. Liu, H. Ye, C. W. Tan, and K.-Y. Lam, "Efficient federated unlearning with adaptive differential privacy preservation," in 2024 IEEE International Conference on Big Data (BigData). IEEE, 2024, pp. 7822–7831.
- [33] V. S. Chundawat, P. Niroula, P. Dhungana, S. Schoepf, M. Mandal, and A. Brintrup, "Conda: Fast federated unlearning with contribution dampening," arXiv preprint arXiv:2410.04144, 2024.
- [34] C. Wu, S. Zhu, P. Mitra, and W. Wang, "Unlearning backdoor attacks in federated learning," in 2024 IEEE Conference on Communications and Network Security (CNS). IEEE, 2024, pp. 1–9.
- [35] X. Sheng, W. Bao, and L. Ge, "Robust federated unlearning," in *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, 2024, pp. 2034–2044.
- [36] Z. Liu, Y. Jiang, W. Jiang, J. Guo, J. Zhao, and K.-Y. Lam, "Guaranteeing data privacy in federated unlearning with dynamic user participation," *IEEE Transac*tions on Dependable and Secure Computing, 2024.
- [37] T. T. Huynh, T. B. Nguyen, P. L. Nguyen, T. T. Nguyen, M. Weidlich, Q. V. H. Nguyen, and K. Aberer, "Fast-fedul: a training-free federated unlearning with provable skew resilience," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2024, pp. 55–72.
- [38] W. Su, P. Wang, Z. Wang, A. Muhammad, T. Tao, X. Tong, and Q. Zhang, "F2ul: Fairness-aware federated unlearning for data trading," *IEEE Transactions on*

- Mobile Computing, 2024.
- [39] J. Chen, Q. Hu, F. Zhong, Y. Zhuang, and M. Xu, "Upcycling noise for federated unlearning," *arXiv preprint arXiv:2412.05529*, 2024.
- [40] Z. Deng, L. Luo, and H. Chen, "Enable the right to be forgotten with federated client unlearning in medical imaging," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2024, pp. 240–250.
- [41] Z. Pan, Z. Wang, C. Li, K. Zheng, B. Wang, X. Tang, and J. Zhao, "Federated unlearning with gradient descent and conflict mitigation," arXiv preprint arXiv:2412.20200, 2024.
- [42] W. Wang, C. Zhang, Z. Tian, and S. Yu, "Fedu: Federated unlearning via user-side influence approximation forgetting," *IEEE Transactions on Dependable and Secure Computing*, 2024.
- [43] G. Li, X. Feng, L. Wang, H. Wu, and B. Düdder, "Federated unlearning in the internet of vehicles," in 2024 54th Annual IEEE/IFIP International Conference on Dependable Systems and Networks-Supplemental Volume (DSN-S). IEEE, 2024, pp. 96–103.
- [44] Z. Ma, H. Tu, L. Zhou, X. Yan, H. Xu, Z. Wang, S. Chen *et al.*, "Hier-fun: Hierarchical federated learning and unlearning in heterogeneous edge computing," *IEEE Internet of Things Journal*, 2024.
- [45] X. Yu, W. Pei, B. Xue, and Q. Zhang, "Federated unlearning model recovery in data with skewed label distributions," *arXiv preprint arXiv:2412.13466*, 2024.
- [46] Z. Wang, X. Gao, C. Wang, P. Cheng, and J. Chen, "Efficient vertical federated unlearning via fast retraining," *ACM Transactions on Internet Technology*, vol. 24, no. 2, pp. 1–22, 2024.
- [47] W. Zhu, J. Jia, B. Luo, and X. Lin, "Federated unlearning with multiple client partitions," in *ICC 2024-IEEE International Conference on Communications*. IEEE, 2024, pp. 5190–5195.
- [48] Z. Xie, Z. Gao, Y. Lin, C. Zhao, X. Yu, and Z. Chai, "Adaptive clipping and distillation enabled federated unlearning," in 2024 IEEE International Conference on Web Services (ICWS). IEEE, 2024, pp. 748–756.
- [49] A. Mora, L. Dominici, and P. Bellavista, "Fedunran: On-device federated unlearning via random labels," in 2024 IEEE International Conference on Big Data (BigData). IEEE, 2024, pp. 7955–7960.
- [50] Y. Jiang, C. W. Tan, and K.-Y. Lam, "Feduhb: Accelerating federated unlearning via polyak heavy ball method," in 2024 IEEE Information Theory Workshop (ITW). IEEE, 2024, pp. 235–240.
- [51] A. Mora, L. Valerio, P. Bellavista, and A. Passarella, "Fedquit: On-device federated unlearning via a quasi-competent virtual teacher," arXiv preprint arXiv:2408.07587, 2024.
- [52] Z. Xiong, W. Li, and Z. Cai, "Appro-fun: Approximate machine unlearning in federated setting," in 2024 33rd International Conference on Computer Communications and Networks (ICCCN). IEEE, 2024, pp. 1–9.
- [53] Y. H. Khalil, L. Brunswic, S. Lamghari, X. Li,

- M. Beitollahi, and X. Chen, "Not: Federated unlearning via weight negation," *arXiv preprint arXiv:2503.05657*, 2025.
- [54] M. Ameen, P. Wang, W. Su, X. Wei, and Q. Zhang, "Speed up federated unlearning with temporary local models," *IEEE Transactions on Sustainable Computing*, 2025.
- [55] J. Zhang, M. Zhao, Z. Wang, W. Su, and P. Wang, "Model recovery in federated unlearning with restricted server data resources," *IEEE Internet of Things Journal*, 2025.
- [56] W. Wang, Q. Ma, Z. Zhang, Y. Liu, Z. Liu, and M. Fang, "Poisoning attacks and defenses to federated unlearning," *arXiv preprint arXiv:2501.17396*, 2025.
- [57] T. T. Huynh, T. B. Nguyen, T. T. Nguyen, P. L. Nguyen, H. Yin, Q. V. H. Nguyen, and T. T. Nguyen, "Certified unlearning for federated recommendation," *ACM Transactions on Information Systems*, vol. 43, no. 2, pp. 1–29, 2025.
- [58] X. Gao, X. Ma, J. Wang, Y. Sun, B. Li, S. Ji, P. Cheng, and J. Chen, "Verifi: Towards verifiable federated unlearning," *IEEE Transactions on Dependable and Secure Computing*, 2024.
- [59] W. Yuan, H. Yin, F. Wu, S. Zhang, T. He, and H. Wang, "Federated unlearning for on-device recommendation," in *Proceedings of the 16th ACM International Conference on Web Search and Data Mining*, 2023, pp. 393–401.
- [60] R. Jin, M. Chen, Q. Zhang, and X. Li, "Forgettable federated linear learning with certified data removal," *arXiv preprint arXiv:2306.02216*, 2023.
- [61] Y. Li, C. Chen, X. Zheng, and J. Zhang, "Federated unlearning via active forgetting," *arXiv preprint* arXiv:2307.03363, 2023.
- [62] Y. Zhao, P. Wang, H. Qi, J. Huang, Z. Wei, and Q. Zhang, "Federated unlearning with momentum degradation," *IEEE Internet of Things Journal*, 2023.
- [63] X. Guo, P. Wang, S. Qiu, W. Song, Q. Zhang, X. Wei, and D. Zhou, "Fast: Adopting federated unlearning to eliminating malicious terminals at server side," *IEEE Transactions on Network Science and Engineering*, 2023.
- [64] P. Wang, Z. Wei, H. Qi, S. Wan, Y. Xiao, G. Sun, and Q. Zhang, "Mitigating poor data quality impact with federated unlearning for human-centric metaverse," *IEEE Journal on Selected Areas in Communications*, 2023.
- [65] J. Gong, J. Kang, O. Simeone, and R. Kassab, "Forget-svgd: Particle-based bayesian federated unlearning," in 2022 IEEE Data Science and Learning Workshop (DSLW). IEEE, 2022, pp. 1–6.
- [66] G. Ye, Q. V. H. Nguyen, and H. Yin, "Heterogeneous decentralized machine unlearning with seed model distillation," *arXiv preprint arXiv:2308.13269*, 2023.
- [67] M. Ameen, R. U. Khan, P. Wang, S. Batool, and M. Alajmi, "Addressing unreliable local models in federated learning through unlearning," *Neural Networks*, p. 106688, 2024.

- [68] H. Xu, T. Zhu, L. Zhang, W. Zhou, and P. S. Yu, "Update selective parameters: Federated machine unlearning based on model explanation," *IEEE Transactions on Big Data*, 2024.
- [69] Q. Guo, Z. Tian, M. Yao, Y. Qi, S. Qi, Y. Li, and J. S. Dong, "Forgetting through transforming: Enabling federated unlearning via class-aware representation transformation," *arXiv* preprint arXiv:2410.06848, 2024.
- [70] A. Dhasade, Y. Ding, S. Guo, A.-M. Kermarrec, M. de Vos, and L. Wu, "Quickdrop: Efficient federated unlearning via synthetic data generation," in *Proceedings of the 25th International Middleware Conference*, 2024, pp. 266–278.
- [71] H. Xia, S. Xu, J. Pei, R. Zhang, Z. Yu, W. Zou, L. Wang, and C. Liu, "Fedme 2: Memory evaluation & erase promoting federated unlearning in dtmn," *IEEE Journal on Selected Areas in Communications*, 2023.
- [72] K. ElBedoui, "Ecg classifiction based on federated unlearning," in 2023 International Symposium on Networks, Computers and Communications (ISNCC). IEEE, 2023, pp. 1–5.
- [73] L. Zhou, Y. Zhu, Q. Xue, J. Zhang, and P. Zhang, "Streamlined federated unlearning: Unite as one to be highly efficient," *arXiv preprint arXiv:2412.00126*, 2024.
- [74] W. Wang, Z. Tian, C. Zhang, A. Liu, and S. Yu, "Bfu: Bayesian federated unlearning with parameter self-sharing," in *Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security*, 2023, pp. 567–578.
- [75] M. Alam, H. Lamri, and M. Maniatakos, "Get rid of your trail: Remotely erasing backdoors in federated learning," *arXiv preprint arXiv:2304.10638*, 2023.
- [76] G. Li, L. Shen, Y. Sun, Y. Hu, H. Hu, and D. Tao, "Subspace based federated unlearning," *arXiv* preprint *arXiv*:2302.12448, 2023.
- [77] P. Wang, W. Song, H. Qi, C. Zhou, F. Li, Y. Wang, P. Sun, and Q. Zhang, "Server-initiated federated unlearning to eliminate impacts of low-quality data," *IEEE Transactions on Services Computing*, 2024.
- [78] P. Wang, Z. Yan, M. S. Obaidat, Z. Yuan, L. Yang, J. Zhang, Z. Wei, and Q. Zhang, "Edge caching with federated unlearning for low-latency v2x communications," *IEEE Communications Magazine*, 2023.
- [79] H. Gu, W. Ong, C. S. Chan, and L. Fan, "Ferrari: federated feature unlearning via optimizing feature sensitivity," *Advances in Neural Information Processing Systems*, vol. 37, pp. 24150–24180, 2024.
- [80] N. K. Dinsdale, M. Jenkinson, and A. I. Namburete, "Fedharmony: unlearning scanner bias with distributed data," in *International Conference on Medical Im*age Computing and Computer-Assisted Intervention. Springer, 2022, pp. 695–704.
- [81] Y. Liu, Z. Ma, Y. Yang, X. Liu, J. Ma, and K. Ren, "Revfrf: Enabling cross-domain random forest training with revocable federated learning," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 6, pp. 3671–3685, 2021.

- [82] Z. Deng, Z. Han, C. Ma, M. Ding, L. Yuan, C. Ge, and Z. Liu, "Vertical federated unlearning on the logistic regression model," *Electronics*, vol. 12, no. 14, p. 3182, 2023.
- [83] J. Zhang, B. L. J. Li, and C. Wu, "Securecut: Federated gradient boosting decision trees with efficient machine unlearning," *arXiv preprint arXiv:2311.13174*, 2023.
- [84] M. Han, T. Zhu, L. Zhang, H. Huo, and W. Zhou, "Vertical federated unlearning via backdoor certification," *IEEE Transactions on Services Computing*, 2025.
- [85] Z. Pan, Z. Ying, Y. Wang, C. Zhang, W. Zhang, W. Zhou, and L. Zhu, "Feature-based machine unlearning for vertical federated learning in iot networks," *IEEE Transactions on Mobile Computing*, 2025.
- [86] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings et al., "Advances and open problems in federated learning," Foundations and Trends® in Machine Learning, vol. 14, no. 1–2, pp. 1–210, 2021.
- [87] Z. Xiong, W. Li, Y. Li, and Z. Cai, "Exact-fun: An exact and efficient federated unlearning approach," in 2023 IEEE International Conference on Data Mining (ICDM). IEEE, 2023.
- [88] Y. Tao, C.-L. Wang, M. Pan, D. Yu, X. Cheng, and D. Wang, "Communication efficient and provable federated unlearning," *arXiv preprint arXiv:2401.11018*, 2024.
- [89] J. Shao, T. Lin, X. Cao, and B. Luo, "Federated unlearning: a perspective of stability and fairness," *arXiv* preprint arXiv:2402.01276, 2024.
- [90] S. Wang, B. Liu, and G. Zuccon, "How to forget clients in federated online learning to rank?" in *European Conference on Information Retrieval*. Springer, 2024, pp. 105–121.
- [91] C. Fu, W. Jia, and N. Ruan, "Client-free federated unlearning via training reconstruction with anchor subspace calibration," in *IEEE International Conference* on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2024, pp. 9281–9285.
- [92] K. Chen, Y. Chai, W. Zhang, S. Wang, J. Shen *et al.*, "Federated unlearning for human activity recognition," *arXiv preprint arXiv:2404.03659*, 2024.
- [93] H. Wang, X. Zhu, C. Chen, and P. Esteves-Verssimo, "Goldfish: An efficient federated unlearning framework," in 2024 54th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). IEEE, 2024, pp. 252–264.
- [94] X. Zhu, G. Li, and W. Hu, "Heterogeneous federated knowledge graph embedding learning and unlearning," in *Proceedings of the ACM Web Conference*, 2023, pp. 2444–2454.
- [95] Z. Liu, H. Ye, Y. Jiang, J. Shen, J. Guo, I. Tjuawinata, and K.-Y. Lam, "Privacy-preserving federated unlearning with certified client removal," *arXiv preprint arXiv:2404.09724*, 2024.
- [96] Y. Jiang, J. Shen, Z. Liu, C. W. Tan, and K.-Y. Lam, "Towards efficient and certified recovery from

- poisoning attacks in federated learning," arXiv preprint arXiv:2401.08216, 2024.
- [97] Y. Fraboni, M. Van Waerebeke, K. Scaman, R. Vidal, L. Kameni, and M. Lorenzi, "Sifu: Sequential informed federated unlearning for efficient and provable client unlearning in federated optimization," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2024, pp. 3457–3465.
- [98] Y. Yuan, B. Wang, C. Zhang, Z. Xiong, C. Li, and L. Zhu, "Towards efficient and robust federated unlearning in iot networks," *IEEE Internet of Things Journal*, 2024.
- [99] T.-H. Nguyen, H.-P. Vu, D. T. Nguyen, T. M. Nguyen, K. D. Doan, and K.-S. Wong, "Empirical study of federated unlearning: Efficiency and effectiveness," in *Asian Conference on Machine Learning*. PMLR, 2024, pp. 959–974.
- [100] J. Chen, Z. Lin, W. Lin, W. Shi, X. Yin, and D. Wang, "Fedmua: Exploring the vulnerabilities of federated learning to malicious unlearning attacks," *IEEE Transactions on Information Forensics and Security*, 2025.
- [101] A. Halimi, S. Kadhe, A. Rawat, and N. Baracaldo, "Federated unlearning: How to efficiently erase a client in fl?" *arXiv preprint arXiv:2207.05521*, 2022.
- [102] Y. Liu, Z. Ma, X. Liu, J. Liu, Z. Jiang, J. Ma, P. Yu, and K. Ren, "Learn to forget: Machine unlearning via neuron masking," *arXiv preprint arXiv:2003.10933*, 2020.
- [103] T. Che, Y. Zhou, Z. Zhang, L. Lyu, J. Liu, D. Yan, D. Dou, and J. Huan, "Fast federated machine unlearning with nonlinear functional theory," 2023.
- [104] T. Shaik, X. Tao, L. Li, H. Xie, T. Cai, X. Zhu, and Q. Li, "Framu: Attention-based machine unlearning using federated reinforcement learning," *IEEE Transac*tions on Knowledge and Data Engineering, 2024.
- [105] A. K. Varshney, K. Vandikas, and V. Torra, "Unlearning clients, features and samples in vertical federated learning," *arXiv preprint arXiv:2501.13683*, 2025.
- [106] Y. Lin, Z. Gao, H. Du, J. Ren, Z. Xie, and D. Niyato, "Blockchain-enabled trustworthy federated unlearning," *arXiv preprint arXiv:2401.15917*, 2024.
- [107] J. Wang, S. Guo, X. Xie, and H. Qi, "Federated unlearning via class-discriminative pruning," in *Proceedings of the ACM Web Conference*, 2022, pp. 622–632.
- [108] Y. Lin, Z. Gao, H. Du, D. Niyato, G. Gui, S. Cui, and J. Ren, "Scalable federated unlearning via isolated and coded sharding," arXiv preprint arXiv:2401.15957, 2024
- [109] Y. Lin, Z. Gao, H. Du, D. Niyato, J. Kang, and X. Liu, "Incentive and dynamic client selection for federated unlearning," in *Proceedings of the ACM on Web Con*ference, 2024, pp. 2936–2944.
- [110] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," arXiv preprint arXiv:1806.00582, 2018.
- [111] J. Tang, X. Ding, D. Hu, B. Guo, Y. Shen, P. Ma, and Y. Jiang, "Fedrad: Heterogeneous federated learning via relational adaptive distillation," *Sensors*, vol. 23, no. 14,

- p. 6518, 2023.
- [112] Q. Wang, R. Xu, S. He, R. Berry, and M. Zhang, "Unlearning incentivizes learning under privacy risk," in *THE WEB CONFERENCE* 2025.
- [113] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to {Byzantine-Robust} federated learning," in *29th USENIX security symposium (USENIX Security 20)*, 2020, pp. 1605–1622.
- [114] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5650–5659.
- [115] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," *Advances in neural information processing systems*, vol. 30, 2017.
- [116] N. Ding, Z. Sun, E. Wei, and R. Berry, "Incentive mechanism design for federated learning and unlearning," in Proceedings of the 24th International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing, 2023, pp. 11–20.
- [117] X. Zuo, M. Wang, T. Zhu, L. Zhang, S. Yu, and W. Zhou, "Federated learning with blockchain-enhanced machine unlearning: A trustworthy approach," *IEEE Transactions on Services Computing*, 2025.
- [118] S. Schelter, "amnesia-towards machine learning models that can forget user data very fast," in *1st International Workshop on Applied AI for Database Systems and Applications (AIDB19)*, 2019.
- [119] 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. 10422–10431.
- [120] S. Misra and Y. Wu, "Machine learning assisted segmentation of scanning electron microscopy images of organic-rich shales with feature extraction and feature ranking," *Machine learning for subsurface characterization*, vol. 289, no. 3, p. 4, 2019.
- [121] J. Liu, M. Xue, J. Lou, X. Zhang, L. Xiong, and Z. Qin, "Muter: Machine unlearning on adversarially trained models," in *Proceedings of the IEEE/CVF International* Conference on Computer Vision, 2023, pp. 4892–4902.
- [122] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *International conference on artificial intelligence and statistics*. PMLR, 2020, pp. 2938–2948.
- [123] Z. Zhang, A. Panda, L. Song, Y. Yang, M. Mahoney, P. Mittal, R. Kannan, and J. Gonzalez, "Neurotoxin: Durable backdoors in federated learning," in *International Conference on Machine Learning*. PMLR, 2022, pp. 26429–26446.
- [124] Q. Li, Y. Diao, Q. Chen, and B. He, "Federated learning on non-iid data silos: An experimental study," in 2022 *IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2022, pp. 965–978.

TABLE VII: Key Terminology Related to FL

Terminology	Description
FL	A decentralized ML where multiple devices train a shared model without sharing raw data.
Server	A central entity orchestrates FL by distributing the global model, aggregating updates, and coordinating training.
Client	Individual devices or endpoints participating in FL. Each client possesses local data to train a local model.
Global Model	A model generated by aggregating clients' local model updates.
Local Model	A model trained on individual clients' data, updated locally during FL rounds.
Aggregation	A method to combine local model updates to create a global model
FedAvg	An aggregation method averaging model parameters from participating clients.
Asynchronous FL	Allows clients to update the global model without synchronization at different timings.
Cross-silo FL	A type of FL that trains a global model across 2-100 data silos.
Cross-device FL	A type of FL that trains a global model up to 10 <sup>10</sup> clients (mobile or IoT devices).
VFL	A type of FL that trains a global model across vertically partitioned data sources with different features but shared sample IDs.
HFL	A type of FL that trains a global model across horizontally partitioned data sources with the same features but different samples.
IID data	Data sampled from the identical and independent distribution.
Non-IID data	Data sampled from not identical or not independent distribution across clients. Also known as heterogeneous data.
HE	Homomorphic Encryption. A cryptographic technique that allows computation on the encrypted model parameters.
DP	Differential Privacy. Ensures the presence or absence of an individual's data does not significantly affect the output.

#### APPENDIX

## KEY TERMINOLOGIES RELATED TO FL

Table VII summarizes the key terminologies and descriptions, including FL components, various types of FL, data distribution, and standard privacy-preserving techniques.

## **EVALUATION METRIC**

In Table VIII, we included the reference for each metric. The predominant metric used for evaluating efficacy was accuracy on the target dataset, followed by the attack success rate on backdoor attacks. For assessing fidelity, the majority of works measured accuracy on the test dataset, while efficiency was predominantly evaluated based on time reduction in seconds or the number of rounds.

## COMPREHENSIVE TABLES

Across Table IX to Table XIV, we summarized all the published literature on FU in multiple dimensions of assumptions, unlearning methods, their configurations, and links to the code, if released. It would give the readers a more comprehensive view of all the works and make it easy to compare them.

<sup>&</sup>lt;sup>13</sup>Adult, bank market, drug consumption, wine quality (classification), superconduct, appliance energy, insurance company, news popularity (regression)

<sup>&</sup>lt;sup>15</sup>ImageNet and VGGFace are ImageNet\_mini, VGGFace\_mini

TABLE VIII: A summary of Evaluation Metrics with References

Objective	Category	Metric	Reference
Efficacy	Performance	Accuracy on the target set	[5, 18, 19, 21, 22, 25, 29, 31, 33, 35, 38, 44, 45, 49, 51, 53, 55, 58, 60, 62, 63, 65, 68–70, 73, 74, 79, 85, 92, 97, 100, 101, 107, 117]
		F1 or AUC on the target set	[69]
		Loss or errors on the target set	[5, 40, 58, 103]
		MSE and MAE	[81, 104]
	Difference	L2 distance	[72, 74, 104]
		KLD	[58, 74]
		Error rate (SAPE, ECE)	[26, 27, 46, 52]
		Angular deviation	[2, 37, 57, 99, 106]
		Wasserstein distance	[83]
	Indiscrimination quality	ASR, precision, or recall on BA	[2, 18, 20, 32–35, 37, 41–43, 50, 56, 57, 60–62, 73–76, 82, 84, 91, 93, 95, 96, 99–101, 103]
		ASR, precision, or recall on MIA	[5, 18, 20, 25, 29, 32, 33, 35, 48–54, 68–70, 79, 84, 85, 88, 91, 92, 95, 96, 98, 101, 105–109]
		Model Inversion attack	[68, 79]
		Multi-task learning	[80]
		Influence function	[58]
		Prediction overlap	[51], Grad-CAM [55], attention map [79], activation map [47]
Fidelity	Performance	Accuracy on test set	[2, 5, 19, 21, 23, 24, 26, 30, 32, 34, 35, 38, 40–43, 45–49, 51, 53, 54, 60–62, 64, 66, 67, 69, 72, 78, 79, 82, 83, 88, 91–93, 96–98, 101, 103, 106, 108, 109]
		F1 or AUC on test set	[24]
		Loss or errors on test set	[5, 50, 52, 54, 56, 63, 64, 67, 105, 106, 108]
		Accuracy on remaining dataset	[18–20, 22, 25, 29, 31, 33, 36–38, 40, 42, 44, 51, 53, 55, 60, 62, 68–71, 73, 75–77, 79, 84, 85, 92, 100, 107]
		F1 or AUC on remaining set	[40, 69, 81, 85, 105]
		Loss or errors on remaining set	[21, 40, 55, 77, 85, 103]
Efficiency	Complexity	Time taken (in # rounds, s, min)	[5, 18, 21–25, 27, 29, 30, 33–37, 40–42, 44, 46–50, 52, 54, 57, 58, 61, 63, 66–68, 70, 73, 77, 79, 84, 88, 91, 92, 95–98, 101, 103, 106, 108, 117]
		Speed-up ratio	[2, 62, 69]
		Memory in MB/GB	[24, 25, 36, 37, 48, 57, 58, 63, 68, 81, 95, 108]

TABLE IX: A comprehensive table summarizing the literature on FU. "Remaining Acc." and " Target Acc." are short for accuracy on remaining and target data, respectively. "n/d" represents not disclosed within the paper, and "-" indicates none. The numbers in brackets in the "Data Dist." column indicate the range of the concentration parameter they used. Evaluation metrics for each objective are separated by ";" Note that we added links to the code if they are released.

Ref.	. Unlearner	Target	Influence removal	Performance recovery	Implication	Eval. Metric	Dataset	Data Dist.	Agg. Method	Model Arch.	Code
[81]	Server	Client	Remove the node of the target client and all the child nodes	- (removing only a few (i=5) clients does not influence the efficacy of RevFRF)	Efficacy, Fidelity, Security, Guarantee	MSE, MAE, R-Square; Remaining Acc., recall, F1; time(s), memory(MB)	UCI tabular datasets <sup>13</sup>	p/u	p/u	Random	1
[87]	[87] All clients	Sample	Retrain the remaining clients (exact unlearning) using a quantized learning model		Efficacy, Fidelity, Efficiency	ASR (MIA), SAPE on the unlearned model; Speed up ratio	fMNIST, CIFAR10	Non-IID (random)	FedAvg	3, 4-layer CNN	[link]
[46]	[46] Server	Feature	Retrain from historically saved local model checkpoints	Automatic optimizer control for generalization	Efficacy, Efficiency, Guarantee, Adaptivity	SAPE; Test Acc.; Speed-up ratio	MNIST, CIFAR10, Speechcommand, ImageNet, BrainTumorMRI, ModelNet	p/u	p/u	LeNet5, ResNet18, M5, ResNet34, pointConv	
[06]	Remaining clients	Client	Calibrate the local update using the stored historical local updates' direction	·	Efficacy, Fidelity	NDCG@10	MQ2007, MSLR-WEB10k, Yahoo, Istella-S		FedAvg	Linear model	[link]
[2]	Server, Remaining clients	Client	Recover from historically saved remaining local model updates	Pre- and Post-training: the remaining clients train and send the updates in the first and last several rounds	Efficacy, Efficiency, Security	ASR (BA); Test error; time saved % (# rounds)	MNIST, fMNIST, Purchase, HAR	Non-IID (Fang [0.1,1])	FedAvg, median, trimmed- mean	3-layer CNN, FCNN	ı
[34]	Server	Sample	Recover from historically saved remaining local model updates by averaging them and subtracting the target clients' update	KD: using the old global model as a teacher model, transfer remembered knowledge to the skewed unlearning model (student model)	Efficacy, Fidelity	ASR (BA); Test Acc	MNIST, CIFAR10, GTSRB	p/u	FedAvg	2-layer CNN, VGG11, AlexNet	
[29]	[29] Server	Sample	Recover from historically saved local model updates by removing a weighted sum of them from the global model	Add Gaussian noise for indistinguishability of the retrained and unlearned model	Efficacy, Fidelity, Efficiency, Security, Guarantee	Target Acc., ASR (MIA); Remaining Acc.; time (s)	MNIST, CIFAR10, SVHN, USPS	(II)	FedAvg	pre-trained CNN	1
[82]	Server	Client, Feature	Recover from saved target model updates at the last round by subtracting them from the global model	Constraints on intermediate local model parameter during unlearning	Efficacy, Fidelity, Efficiency, Security	ASR (data poisoning); acc on test dataset; time (# rounds)	Cood-RNA, Iris, Adult, Breast- cancer	OII	p/u	CNN	1
[64]	[64] Target Client	Sample, Client	Recover from historically saved updates and subtract the target model updates	Fine-tune the unlearned model using the direction of the target updates	Efficiacy, Fidelity, Efficiency, Adaptivity	MAPE, Test Acc. and Loss	Pix3D, VRHP	IID, Non-IID (Dirichlet 0.1)	FedAvg	VGG16, LSTM	
[5]	Remaining clients	Client	Recover from periodically saved local model updates by applying weighted averages to them	Post-training: a few training rounds after unlearning to redirect the updates without the target client	Efficiency, Fidelity,	Prediction difference, attack precision and recall (MIA), Target Acc., loss; Remaining Acc., loss; time(s)	Adult, Purchase, MNIST, CIFAR10	p/u	FedAvg	2-, 3-layer NN, 4-layer CNN	[link]

TABLE X: A comprehensive table summarizing the literature on FU. (continued)

		[link]	,	[link]	, AUGUST 20			ı	ı	1	[link]	,	ı		24
NCF,	LightGCN	Logistic   regression model, CNN	p/u	3-layer CNN, ResNet18	CNN, LeNet, ResNet18	4-layer CNN	DNN	ResNet18, ResNet50	p/u	LeNet5, ResNet18, CNN- LSTM, DenseNet121	ResNet,   pre-trained VGG	GBDT	MLP, 2-layer CNN, VGG11, MobileNet	3-layer CNN	3-layer CNN
	FedAvg	FedAvg	p/u	FedAvg	FedAvg	Avg. base layers	FedAvg	FedAvg	FedAvg	FedAvg, Krum, Median	FedAvg	p/u	FedAvg	FedAvg	FedAvg
	p/u	IID, Non-IID (Dirichlet $\alpha = 1$ )	p/u	OII	p/u	Non-IID	Non-IID (random)	Non-IID (Dirichlet)	Non-IID (concept drift)	IID, Non-IID (Dirichlet [0.5, 0.9]	IID, Non-IID (Fang)	p/u	IID, Non-IID (random)	OII	
	MovieLens-100k, Steam-200k	MNIST, CIFAR, CelebA	MNIST, CIFAR10, SVHN	MNIST, CIFAR10, CIFAR100	fMNIST, CIFAR10, SVHN	MovieLens-100k	MNIST	MNIST, fMNIST, CIFAR10, CIFAR100	AMPds2, METR- LA, MIMIC-III, NYPD, MIMIC- CXR, SHED	MNIST, CIFAR10, SpeechCommand, ISIC, COVID, ImageNet, VGGFace <sup>9</sup>	CIFAR10, CIFAR100	Give credit, optDigits, Epsilon	MNIST, fMNIST, CIFAR10, SVHN	MIT-BIH, BIDMC	MNIST, ÍMNIST
, ( <u> </u>	HR@10, NDCG@10	Target Acc.; time (#rounds)	Prec., Recall (MIA); Test Acc., Time (s)	ASR (BA), Cosine simm.	Errors on target data; errors on remaining/test data, Test Acc.; time (s)	Test Acc.	Target Acc.; Remaining Acc.	Target Acc., ASR (BA); Remaining Acc.; speed up ratio	MSE, MAE, L2 distance	Target Acc. and loss, IF, KLD; time (s), space (MB)	Target Acc., ASR (MIA); Remaining Acc.; speed up ratio	1st Wasserstein distance; Test Acc.	Target Acc., F1 score; Test Acc., loss; time(CPU, s), memory(MB)	L2 distance in confidence; Test Acc., loss	Target Acc., ASR (MIA, BA); Remaining Acc., Test Acc.
	Efficiency, Fidelity, Efficiency, Scalability	Efficacy, Fidelity, Efficiency, Security, Guarantee	Efficacy, Fidelity, Efficiency, Adaptivity	Efficacy, Fidelity	Efficiency, Fidelity, Efficiency, Guarantee	Efficacy	Efficacy, Efficiency, Security	Efficiency, Fidelity, Efficiency	Efficacy, Fidelity, Adaptivity	Efficiency, Fidelity, Efficiency, Security	Efficiency, Fidelity, Efficiency	Efficacy, Fidelity, Security	Efficiency, Fidelity, Efficiency, Security	Еfficacy	Efficacy, Fidelity
Gradiant manipulation: calibrate	Grautent manipulation. Cariotate the unlearning model to the historical model updates' direction	•			1	SGD to minimize the impact on the model accuracy	•	Guided fine-tuning on remaining data points' direction		1	Fine-tuning: few FL training with remaining clients without regularization	Post-training: a few training rounds after unlearning	Fine-tuning with a small (1/8) sized IID benchmark dataset	-	
D	restore certain global model updates to roll back to the point before the target client joined	Restore and perturb the global model of an optimal iteration	Adjust direction from historically saved local updates only if the model performance drops	Perturb target label to the fake one	Perturbation (Gaussian noise) to the target client	Perturbing target update by generating a reverse gradient	Perturbing target updates by training it on noised input	Scale down on the target data points gradually using the momentum technique	Scale up/down using an attention mechanism to assign reduced weights to the target data	Scale up/down on remaining/target clients' updates	Prune the channel of gradients corresponding to the target	Prune the nodes of the target's local model and retrain it	Subtract target updates from the global model	Subtract target model updates computed only using the target data from the global model	Loss function approximation using Hessian matrix calculated on data the server owns
	Client	Client	Client	Client	Sample	Sample	Client	Class, Client	Sample	Client	Class	Sample, Feature	Client	Sample, Client	Client
	Server, All clients	7] Server, Remaining clients	8] Server	Server	[103]Target client	[78] Server	[31] Target client	Server, target client	[104] All clients	[58] Server	[107] Server and all clients	[83] Target clients	3] Server	[72] Target client	[60] Target client
	[59]	[67]	[86]	[66]	[1]	[]	[3	[62]		[5	11	8	[63]	[7	9]

TABLE XI: A comprehensive table summarizing the literature on FU. (continued)

		BIEX CLASS		14, NO. 8, AU	JGUST 2021		ı			_				25
Code	[link]	1	[link]	[link]	1	1	1	[link]	,	[link]	ı		1	1
THORE WE	3-layer CNN, AlexNet, ResNet	MobileNetV3, ResNet, RegNet <sup>11</sup>	3-layer BNN, ResNet18	VGG-based CNN	3-layer ConvNet	1-layer BNN	NN	VGG16, LeNet5, MLP, GPT2	p/u	TransE, ComplEx, RotE	3-layer CNN, ResNet10	ResNet, MobileNet	p/u	MLP, 3-layer CNN, ResNet18
Method	FedAvg	FedAvg	FedAvg	FedEqual	FedAvg	p/u	FedAvg	FedAvg, FedBuff	FedAvg	FedAvg	FedAvg	p/u	FedAvg	p/u
Data Dist.	OII OII	p/u	p/u	Non-IID (covariate shift)	IID, Non-IID (Dirichlet $\alpha = 0.1$ )	Non-IID (unique)	Non-IID (unique)	Non-IID (Dirichlet $\alpha = 5$ )	IID, Non- IID	Non-IID (unique)	p/u	Non-IID (unique)	IID, Non- IID (unique)	IID, Non-IID (Dirichlet [0.1, 0.6])
Dataset	MNIST, fMNIST, CIFAR10, CelebA	SVHN, CelebA	MNIST, CIFAR10	ABIDE	MNIST, CIFAR10, SVHN	MNIST	fMNIST	CIFAR10, EMNIST, Purchase 100, Tiny Shakespeare	MNIST, fMNIST, CIFAR10	FB15k-237	MNIST, CIFAR10, CelebA	MNIST, FMNIST, CIFAR10	MNIST, CIFAR10	MNIST, CIFAR10, CIFAR100
Eval. Metric	SAPE; Remaining Acc.; time (s)	Target Acc.; Remaining Acc.	ASR (BA), L2 norm distance, KLD; Test Acc.; time (s)	MAE, SCA	Target Acc., Target ASR (MIA); Remaining ASR (MIA); time (s)	variational posterior, Target Acc.	Target Acc., ECE; Remaining Acc.	Test Acc.; time (s)	F1 (MIA); Test Acc.	Average Hits@N, MRR	ASR (BA); Test Acc.; time(s)	Test Acc., time (min)	Target Acc.; Remaining and Test Acc.	ASR (BA); Test Acc.
Implication	Efficacy, Fidelity, Efficiency, Guarantee, Flexibility	Efficacy, Efficiency, Guarantee	Efficiency, Fidelity, Efficiency	Еfficacy	Efficiency, Fidelity, Efficiency, Scalability	Efficacy, Fidelity	Efficiency, Fidelity, Efficiency	Efficacy, Fidelity Efficiency, Scalability	Efficacy, Fidelity, Fairness	Efficiency, Fidelity, Efficiency	Efficacy, Fidelity, Efficiency	Efficacy, Scalability, Flexibility	Efficacy, Fidelity	Efficacy, Fidelity, Security
Performance recovery	Apply momentum techniques to the Hessian Diagonal	Add a regularization term: L2 norm between the local and global models	Multi-task: The target client performs multi-task learning (maintaining acc on the remaining set)	Multi-task learning: remembrance Efficacy using main task loss for classification	Fine-tuning the unlearned model by including a few original data samples into the training dataset	Post-training: a few training rounds after unlearning	Post-training: a few training rounds after unlearning with remaining clients		Dynamically select remaining clients possessing unbalanced local data	Fine-tuning: passive decay by suppressing the activation of the target data	EWC training, using a Hessian matrix as a regularizer		EWC training, limit the magnitude of parameters through FIM as a regularizer	Server projecting target updates into the orthogonal subspace calculated from remaining clients' representation matrices
Influence removal	Loss function approximation relying on Hessian using diagonal FIM	Multi-task learning: data erasure and remembrance by optimizing local model loss using MIA-like evaluation model	Multi-task: The target client performs multi-task learning (data erasure).	Multi-task learning: data erasure using losses to discriminate between the sites and to remove site-specific information	Reverse training (stochastic gradient ascent) on the distilled dataset that each client has created	Variational inference, maximizing local loss	Variational inference, quantization and sparsification method to reduce communication overhead	Cluster clients based on their similarity (training time and low model disparity) and retrain the only cluster that contains the target client.	Cluster based on their similarity and calibrate the gradient by step length and direction	Knowledge distillation: transfer knowledge to be remembered	Knowledge distillation: the target client's student model distills knowledge from manipulated data (fake labels with original features) that teacher models generated.	Ensemble remaining clients' knowledge-distilled reference models	Reverse training: stochastic gradient ascent	Reverse training: Gradient Ascent (GA)
Target	Client	Feature	Sample	Feature	Class, Client	Sample	Sample, Class, Client	Client	Client	Sample	Sample, Class, Client	Class	Sample, Class, Client	Sample
Unlearner	Target client	[71] All clients	Target client	[80] All clients	All clients	[65] Target client	Target client	Server, Target client, Some remaining clients	[109]Server	Server, All clients	Target clients	Remaining clients	[19] Target client	Server, All clients
Ref.	[27]	[71]	[74]	[80]	[70]	[65]	[26]	[30]	[109]	[94]	[61]	[99]	[19]	[76]

TABLE XII: A comprehensive table summarizing the literature on FU. (continued)

Reverse Training Layer-wise GA Post-training Efficiency, Fidelity, Restraining Light-rouning client updates a foot-training Light-rouning client updates a foot-training Light-rouning the most influential to minimize countriefly and are training the most influential to minimize countriefly and are training to make it Post-training to make it Pos		Target	Influence removal	Performance recovery	Implication	Eval. Metric	Dataset	Data Dist.	Agg. Method	Model Arch.	Code
Post-training Efficacy, Fidelity, Test ACC, Test loss; MNIST, MNIST, Non-IIID ndd Clebch Cleb	Client		Recover from historically saved emaining client updates	l <del> </del>		ASR (BA); Test Acc.; Time (s)	MNIST, CIFAR10, CTSRB, IMDb	IID, Non-IID (Dirichlet [0.1, 1])	FedAvg	3-layer CNN, VGG9, 16, ResNet, MobileNetv2, AlexNet	
Retraining GA   Post-training   Efficiency, Fidelity   Remaining Acc.   MNIST, Nov-IID   FedAvg	Server, Sample Target client		Reverse Training: Layer-wise GA	Post-training	Efficacy, Fidelity, Efficiency, Adaptivity	7.5	MNIST, fMNIST, CIFAR10, CelebA	Non-IID	p/u	LeNet5, ResNet18	
Retrain the low-performing cluster - Efficiency, Fields By, and a continuous cluster - Efficiency, Fields By, and a continuous cluster between - Fields By, again and by a continuous conti	Sample	1	Reverse Training: GA	Post-training	Efficacy, Fidelity	Remaining Acc., loss; Time (s)	MNIST, fMNIST, CIFAR10, CelebA	Non-IID	FedAvg	MLP, LeNet5, ResNet18, MobileNet	
Subtract differences between traget model training the most influenting to make it posterining to most influenting to make it posterining to be forgulated optimization to make it posterining to make it forences training to make it posterining to low-frequency Efficiency, Fidelity, Guarantee (Sch. MED) MED) MED) MED) MED) MED) MED) MED)	Client	1	Retrain the low-performing cluster		Efficacy, Fidelity, Efficiency, Guarantee, Privacy	Remaining Acc.; Time (s), Comm (MB)	MNIST, CIFAR10	p/u	SecAgg	2-layer CNN, ResNet18	
Reverse training: GA Post-training BEfficiency, Fairness Remaining Acc., GIPARIO MIST, IMD Non-IMD Non	Client		Subtract differences between target model updates and the aggregation of selectively stored local updates		Efficacy, Efficiency, Fidelity, Guarantee	ASR (BA), Cosine Similarity; Remaining Acc.; Time (s), Mem (gb)	T, ISTv2	Non-IID (Dirichlet)	FedAvg	p/u	link
res Guided optimization to minimize - Efficiency, Guarantee Acc., Artendron (CIPARIO), Non-IID (CIPARIO) (Dirichlet I, Acc., Test Ac	Server, Client Target client		Reverse training: GA	Post-training	Efficacy, Fidelity, Efficiency, Fairness	Target Acc.; Remaining Acc., Test Acc.; Time (s,	MNIST, fMNIST, KMNIST, CIFAR10	IID, Non- IID	FedAvg	LeNet, AlexNet	1
Finetuning the most influential - Efficiency delay, Fidelity, Acc., Grad Inversion CIFAR10  class channel identifying the target channel identifying the target class channel identifying the retaining set at feature level and components components components channel in the retaining set at feature level and components components channel identifying and XOR with craining and XOR with channel identifying channel identifying and XOR with channel identifying channel identifying and XOR with channel identifying channel identifying channel identifying and XOR with channel identifying channe	Server, Feat Target Client	ures			Efficiency, Fidelity, Efficiency, Guarantee	ASR (MIA), Target Acc., Attention map; Remaining Acc., Test Acc; Time (s, Flops)		IID, Non-IID (Dirichlet 1, 10)	p/u	ResNet18, BERT	link
Contrastive unlearning to make it Post-training on low-frequency Efficiency Fidelity, similar to a model trained only on components the retaining set at feature level to be forgotten to be	Class	s	Finetuning the most influential channel identifying the target class		Efficiency, Fidelity,	ASR (MIA), Target Acc., Grad Inversion Attack; Target Acc., Remaining Acc.; Time (s, round), Mem (MB)	MNIST, fMNIST, CIFAR10, CIFAR100	IID, Non-IId	FedAvg	ResNet20, VGG11	1
KD: negative transfer knowledge KD: transfer retaining knowledge Efficiency, Guarantee Acc.; Remaining CIFAR10, Non-IID Random)  Acc.; Time (round) DBpedia (Random)  Acc.; Time (round) DBpedia (Random)  CIFAR10, Non-IID (Random)  Acc.; Time (round) DBpedia (Random)  CIFAR10, Non-IID (Dirichlet 1, Remaining Acc.; CIFAR10, Non-IID (Dirichlet 1, Time)  Reverse training: GA to Post-training with gradient projection at projection given by a projection projection at the projection and contract to the projection and contract training and contract training and contract training and xOR with the local model upon unlearning and xOR with local local model upon unlearning and xOR with local loc	[40] Target Client Class	_ x	Contrastive unlearning to make it similar to a model trained only on the retaining set at feature level	Post-training components	Efficacy, Fidelity, Efficiency	Target Error; Remaining Fl, Acc, Error, Test Acc, Error; Time (s)	RSNA-ICH, ISIC2018	Non-IID (Dirichlet 1)	FedAvg	DenseNet121	link
Add auxiliary unlearning module while training and XOR with the local model upon unlearning request the local model upon unlearning projection projection and steepest descent projection and training with gradient projectio	Sam	ple	KD: negative transfer knowledge to be forgotten			ASR (BA), Target Acc.; Remaining Acc.; Time (round)	CIFAR10, CIFAR100, DBpedia	IID, Non-IID (Random)	FedAvg	ResNst18, ResNet44, LSTM	1
Reverse training: GA to Post-training with gradient Efficiency, Fidelity, ASR (BA); Test MNIST, fIID, FedAvg orthogonal steepest descent projection Efficiency, Guarantee Acc.; Time (round) CIFAR10, Non-IID direction (Unique)	Server, Sam Target Client Clas	ple, s, nt	Add auxiliary unlearning module while training and XOR with the local model upon unlearning request	-	Efficiency, Fidelity, Efficiency, Guarantee	ASR (BA, MIA), Target Acc.; Remaining Acc.; Time	MNIST, CIFAR10, CIFAR100	IID, Non-IID (Dirichlet 1, 10)	p/u	LeNet, AlexNet, ResNet18	link
	Clie	nt	Reverse training: GA to orthogonal steepest descent direction		Efficacy, Fidelity, Efficiency, Guarantee		MNIST, fMNIST, CIFAR10, CIFAR100	IID, Non-IID (Unique)	FedAvg	LeNet5, MLP, 2-layer CNN, NFResnet18	

TABLE XIII: A comprehensive table summarizing the literature on FU. (continued)

Ref.	Unlearner	Target	Influence removal	Performance recovery	Implication	Eval. Metric	Dataset	Data Dist.	Agg. Method	Model Arch.	Code
[42]	Target client Class	Class	Approximate influence using Hessian-vector product and subtract it from a global model	local post-training as an adaptive optimization	Efficacy, Fidelity, Efficiency, Guarantee	ASR (BA); Remaining Acc., Test ACC., Time (s)	MNIST, CIFAR10, STL10	OII OII	p/u	5-layer CNN, ResNet18	
[20]	Target client	Sample, Client, Feature	Create a fake dataset to confuse the model and penalize correct prediction by adding reg term	Update salient weights only	Efficacy, Fidelity, Fairness	ASR (MIA, BA); Remaining Acc.	MNIST, CIFAR10, Adult	p/u	p/u	LeNet5, ResNet18, MLP	1
[43]	Server, Target client	Client	Compute Approx. Hessian matrix to estimate recovered gradient	Gradient clipping	Efficacy, Fidelity, Adaptivity	ASR (BA); Test Acc.	MNIST, GTRSB	p/u	FedAvg	CNN	
[21]	Remaining clients	Sample, Class, Client	Rollback before the target join	Retraining in other clients	Efficacy, Efficiency, Security	Target ACC., loss; Test ACC, Remaining Acc; Time (s)	MNIST, CIFAR10	p/u	p/u	p/u	1
[44]	[44] Target client	Client	Clustering, GA on target device	Post-training	Efficacy, Fidelity, Efficiency, Guarantee, Scalability	Target ACC; Test Acc, Remaining ACC, Time (s)	fMNIST, CIFAR10, CIFAR100	Non-IID (Random)	FedAvg	MLP, AlexNet, ResNet18	
[45]	Server, Remaining client	Client	Oversampling from remaining clients to address bias, removing noise from the generated data	Post-training on the oversampled data	Efficacy, Fidelity, Fairness	Target Acc; Test Acc	MNIST, fMNIST, USPS	Covariate shift	Weighted Avg	MLP, 4-layer CNN	1
[69]	All clients	Class	Multi-task learning; change the class of the target sample	Post-training: fine-tuning with the new dataset	Efficiency, Fidelity, Efficiency, Privacy	Target Acc, F1, ASR (MIA); Remaining Acc, F1, Test Acc; Speedup ratio	CIFAR10, CIFAR100, fMNIST, EuroSAT	IID, Non-IID (Dirichlet 0.1, 0.5)	p/u	ResNet18, ResNet50	1
[47]	Server	Client	Client partitioning based on their distribution, remove the subset model	-	Efficacy, Fidelity, Activation mappin Efficiency, Scalability test Acc, Time (s)	Activation mapping; test Acc, Time (s)	MNIST, fMNIST, CIFAR10, SVHN	1	Selective Avg	3-layer CNN, AlexNet, ResNet50, VGG16	1
[22]	Target client	Class, Client	Reverse Learning (GA)		Efficacy, Fidelity, Efficiency	Target Acc; Remaining Acc; time (s)	fMNIST, CIFAR10	p/u	p/u	AlexNet, ResNet18, MobileNetv2	1
[48]	Server	Client	Compute FIM to approximate the parameter and clip them	KD: Using synthesized pseudo samples obtained from the feature space	Efficacy, Fidelity, Efficacy	MIA Prec., Recall; Test Acc; Runtime (s)	MNIST, CIFAR10, CIFAR100	p/u	p/u	LeNet5, ResNet18, ResNet34	1
[49]	Target Client Client	t Client	Replace the true label in the forget set with random ones	Post-training with the modified dataset	Efficacy, Fidelity, Efficiency	ASR (MIA), Target Acc; Test Acc; Time (round)	fMNIST	Semi Non-IID (Unique)	FedAvg	LeNet	link
[21]		Client	Unlearning using LoRA		Efficiency, Efficacy	Target Acc; time (s)	IMDb, Twitter	Non-IID	p/u	GPT-2	
[50]	Remaining clients	Client	Rapid retraining: accelerating using heavy ball method	-	Efficacy, Fidelity, Efficiency, Guarantee	ASR (MIA, BA); Teset loss, Time (s)	MNIST, CIFAR10	p/u	FedAvg	3-layer CNN	
[51]	Target client Client	Client	KD: transfer random prediction on forget data	KD: transfer retaining knowledge Efficacy, Fidelity	Efficacy, Fidelity	ASR (MIA), Target Acc; Remaining Acc, Test Acc, prediction overlap	CIFAR10, CIFAR100, Non- IID (Dirichlet 0.1)	p/u	ResNet18		
[52]	Target client Client	Client	Compute approx hessian with remaining clients		Efficacy, fidelity, Efficiency	ASR (MIA), SAPE; Test loss, Model difference	fMNIST, CIFAR10	p/u	p/u	3-, 4-layer CNN	,

TABLE XIV: A comprehensive table summarizing the literature on FU. (continued)

KNAL	OF BIEX CLA	35 FILES, VOL	1	UGUST 2021	I.	ì	j .	1	ı	İ	28
Code		1	link		1	1		link	1	1	1
Model Arch.	ViT-b/16, 2-layer CNN, ResNet18	3-layer CNN, VGG9, 16, ResNet, MobileNetv2, AlexNet	CNN, AlexNet link	LeNet5, ResNet18, ResNet50	3-layer CNN	MLP, ResNet18	LeNet, ResNEt18, SimpleViT	p/u	CNN, Shallow CNN	ResNet18, LeNet5, 4-layer CNN	LightGCN, NCF
Agg. Method	FedAvg	FedAvg	p/u	FedAvg	FedAvg, Median, Trmean	p/u	FedAvg	p/u	Weighted Avg	FedAvg	FedAvg
Data Dist.	IID, Non-IID (Dirichlet 0.1)	IID, Non-IID (Dirichlet [0.1, 1])	p/u	IDD, Non-IID (Dirichlet 0.1)	Non-IID (Dirichlet 0.5)	p/u	Non-IID (Dirichlet 1, 0.1)	p/u	IID, Non-IID (Random)	IID, Non-IID (Dirichlet 0.5)	Covariate shift
Dataset	Caltech101, CIFAR10, CIFAR100	MNIST, CIFAR10, CTSRB, IMDb	MNIST, fMNIST, n/d CIFAR10	MNIST, SVHN, CIFAR10, CIFAR100	MNIST	UCI, CIFAR10, STL10	fMNIST, CIFAR10, CIFAR100	Adult, Credit, Diabetes, Nursery, Malware	Bank, Credit	Purchase, MNIST, CIFAR10, CIFAR100, Credit	Movielens-Im, Pinterest
Eval. Metric	Target Acc, ASR (MIA); Remaining Acc, Test Acc; Time (flops, bytes)	ASR (BA); Test Acc.; Time (s)	Target Acc, ASR (BA), Recall (MIA); Remaining Acc.; time (s)	Target Acc, Remaining Acc, loss, prediction probs on test set, Grad Cam	Poisoning ASR, Test Error	ASR (MIA); test loss, Remaining F1, AUC; Mem in GB	Target Acc, ASR (MIA); Test Acc, Remaining Acc; Time (round), Mem (byte)	Test Acc, AUC; Time (epoch), Mem (Byte)	Target ACC, ASR (MIA); Remaining F1, Acc; Time (epoch)	Target ACC, ASR (MIA), Remaining ACC	ASR (Poisoning); HR@10, NDCG@10, ER@10, Cosine Sim., Time (s), Mem (gb)
Implication	Efficacy, Fidelity, Efficiency	Efficacy, Fidelity, Efficacy	Efficacy, Fidelity, Efficiency	Efficiency	Efficacy, Fidelity, Security, Guarantee	Efficacy, Fidelity, Efficiency	Efficiency, Fidelity, Efficiency, Guarantee	Adaptivity, Privacy, Fidelity, Efficiency, Guarantee	Efficacy, Fidelity, Ef- ficiency, Guarantee, Adaptibility	Efficacy, Fidelity	Efficacy, Fidelity, Guarantee
Performance recovery	Post-training	KD: transfer remembered knowledge	Post-training	distill only retained knowledge		·	-	·	Gradient alignment	-	
Influence removal	Subtract layer-wise parameters of the global model	Recover from historically saved remaining client updates	Reverse training: GA guided by Avged clean clients	Clip-guided few shot KD using a global model as a student	Compute approx. hessian matrix using limited historical data	GA to unlearn sample, KD to unlearn features and clients	Identify critical layer and mask it out	Subtract target confidence from the original confidence matrix	Dynamically add perturbation to target client updates	Identify influential samples, modify the prediction for the target samples	Rollback, Recover from sampled historical updates
Target	Class	Client	Client	Client	Client	Sample, Client, Feature	Sample, Class, Client	Sample, Feature	Feature	Client	Client
f. Unlearner	Server	[34] Server (	[84] Target client	Server	Server	[105]All clients	Server, Remaining clients	Server	Server	[100]Target Clients	[57] Server (
Ref.	[53]	[34	[84	[55]	[56]	[10	[25]	[24]	[85]	[10	[57