A Survey of Machine Unlearning

Thanh Tam Nguyen[1], Thanh Trung Huynh[2], Zhao Ren[3], Phi Le Nguyen[4], Alan Wee-Chung Liew[1], Hongzhi Yin[5], Quoc Viet Hung Nguyen[1]
[1] Griffith University, [2] École Polytechnique Fédérale de Lausanne, [3] University of Bremen,
[4] Hanoi University of Science and Technology, [5] The University of Queensland

ABSTRACT

Today, computer systems hold large amounts of personal data. Yet while such an abundance of data allows breakthroughs in artificial intelligence, and especially machine learning, its existence can be a threat to user privacy, and it can weaken the bonds of trust between humans and AI. Recent regulations now require that, on request, private information about a user must be removed from both computer systems and from machine learning models – this legislation is more colloquially called "the right to be forgotten"). While removing data from back-end databases should be straightforward, it is not sufficient in the AI context as machine learning models often 'remember' the old data. Contemporary adversarial attacks on trained models have proven that we can learn whether an instance or an attribute belonged to the training data. This phenomenon calls for a new paradigm, namely *machine unlearning*, to make machine learning models forget about particular data. It turns out that recent works on machine unlearning have not been able to completely solve the problem due to the lack of common frameworks and resources. Therefore, this paper aspires to present a comprehensive examination of machine unlearning's concepts, scenarios, methods, and applications. Specifically, as a category collection of cutting-edge studies, the intention behind this article is to serve as a comprehensive resource for researchers and practitioners seeking an introduction to machine unlearning and its formulations, design criteria, removal requests, algorithms, and applications. In addition, we aim to highlight the key findings, current trends, and new research areas that have not yet featured the use of machine unlearning but could benefit greatly from it. We hope this survey serves as a valuable resource for machine learning researchers and those seeking to innovate privacy technologies. Our resources are publicly available at https://github.com/tamlhp/awesome-machine-unlearning.

KEYWORDS

machine unlearning, right to be forgotten, user privacy, decremental learning, certified removal, data forgetting, data deletion, model verification, model repair, model indistinguishability, adversarial attacks

ACM Reference Format:
Thanh Tam Nguyen[1], Thanh Trung Huynh[2], Zhao Ren[3], Phi Le Nguyen[4], Alan Wee-Chung Liew[1], Hongzhi Yin[5], Quoc Viet Hung Nguyen[1] . 2024. A Survey of Machine Unlearning. In *Proceedings of* . ACM, New York, NY, USA, 24 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

, ,
2024. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Computer systems today hold large amounts of personal data. Due to the great advancement in data storage and data transfer technologies, the amount of data being produced, recorded, and processed has exploded. For example, four billion YouTube videos are watched every day [150]). These online personal data, including digital footprints made by (or about) netizens, reflects their behaviors, interactions, and communication patterns in real-world [129]. Other sources of personal data include the digital content that online users create to express their ideas and opinions, such as product reviews, blog posts (e.g. Medium), status seeking (e.g. Instagram), and knowledge sharing (e.g. Wikipedia) [131]. More recently, personal data has also expanded to include data from wearable devices [145]. On the one hand, such an abundance of data has helped to advance artificial intelligence (AI). However, on the other hand, it threatens the privacy of users and has led to many data breaches [12]. For this reason, some users may choose to have their data completely removed from a system, especially sensitive systems such as those do with finance or healthcare [145]. Recent regulations now compel organisations to give users "the right to be forgotten", i.e., the right to have all or part of their data deleted from a system on request [34, 71].

While removing data from back-end databases satisfies the regulations, doing so is not sufficient in the AI context as machine learning models often 'remember' the old data. Indeed, in machine learning systems, often millions, if not billions, of users' data have been processed during the model's training phase. However, unlike humans who learn general patterns, machine learning models behave more like a lossy data compression mechanism [152], and some are overfit against their training data. The success of deep learning models in particular has been recently been attributed to the compression of training data [176, 177]. This memorization behaviour can be further proven by existing works on adversarial attacks [16, 143, 144], which have shown that it is possible to extract the private information within some target data from a trained model. However, we also know that the parameters of a trained model do not tend to show any clear connection to the data that was used for training [162]. As a result, it can be challenging to remove information corresponding to a particular data item from a machine learning model. In other words, it can be difficult to make a machine learning model forget a user's data.

The challenge of enabling users to fully delete their data from a machine learning model has led to the development of a new paradigm: *machine unlearning* [4, 128, 168]. Ideally, a machine unlearning mechanism would remove data from the model without requiring complete retraining [128]. This approach upholds users' right to be forgotten while sparing model owners from frequent and costly retraining.

Table 1: Comparison between existing surveys on machine unlearning

Surveys	Unlearning Framework						Unlearning Scenarios					Unlearning Requests					Applications					
	Definitions	Requirements	Techniques	Verification	Metrics	Resources	Exact	Approximate	Zero-glance	Zero-shot	Few-shot	Item	Feature	Class	Task	Stream	Recommendation	Fed Learning	Graph Embedding	Lifelong Learning	LLM	Generative Models
Ours	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
[185]	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
[183]	✓	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
[160]	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
[152]	✓	✗	✓	✗	✓	✓	✓	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
[122]	✓	✗	✓	✗	✓	✗	✓	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
[204]	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗
[205]	✓	✗	✓	✗	✗	✗	✓	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
[156]	✓	✗	✓	✗	✓	✓	✓	✗	✗	✓	✓	✓	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗
[123]	✓	✗	✓	✗	✓	✓	✓	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗

Further surveys on machine unlearning for specific domains can be found at https://github.com/tamlhp/awesome-machine-unlearning

Researchers have already begun to study aspects of machine unlearning, such as removing part of the training data and analysing the subsequent model predictions [128, 174]. However, it turns out that this problem cannot be completely solved due to a lack of common frameworks and resources [152, 160, 183, 185]. Hence, to begin building a foundation of works in this nascent area, we undertook a comprehensive survey of machine unlearning: its definitions, scenarios, mechanisms, and applications. Our resources are publicly available at [1].

1.1 Reasons for Machine Unlearning

There are many reasons for why a users may want to delete their data from a system. We have categorized these into four major groups: security, privacy, usability, and fidelity. Each reason is discussed in more detail next.

Security. Recently, deep learning models have been shown to be vulnerable to external attacks, especially adversarial attacks [142]. In an adversarial attack, the attacker generates adversarial data that are very similar to the original data to the extent that a human cannot distinguish between the real and fake data. This adversarial data is designed to force the deep learning models into outputting wrong predictions, which frequently results in serious problems. For example, in healthcare, a wrong prediction could lead to a wrong diagnosis, a non-suitable treatment, even a death. Hence, detecting and removing adversarial data is essential for ensuring the model's security and, once an attack is detected, the model needs to be able delete the adversarial data through a machine unlearning mechanism [12, 116].

Privacy. Many privacy-preserving regulations have been enacted recently that involve the right to be forgotten" [10, 34], such as the European Union's General Data Protection Regulation (GDPR) [115] and the California Consumer Privacy Act [134]. In this particular regulation, users must be given the right to have their data and related information deleted to protect their privacy. In part, this legislation has sprung up as a result of privacy leaks. For example, cloud systems can leak user data due to multiple copies of data hold by different parties, backup policies, and replication strategies [163]. In another case, machine learning approaches for genetic data processing were found to leak patients' genetic markers [48, 194]. It is therefore not surprising that users would want to remove their data to avoid the risks of a data leak [12].

Usability. People have difference preferences in online applications and/or services, especially recommender systems. An application will produce inconvenient recommendations if it cannot completely delete the incorrect data (e. g., noise, malicious data, out-of-distribution data) related to a user. For example, one can accidentally search for an illegal product on his laptop, and find that he keeps getting this product recommendation on this phone, even after he cleared his web browser history [12]. Such undesired usability by not forgetting data will not only produce wrong predictions, but also result in less users.

Fidelity. Unlearning requests might come from biased machine learning models. Despite recent advances, machine learning models are still sensitive to bias that means their output can unfairly discriminate against a group of people [120]. For example, COMPAS, the software used by courts to decide parole cases, is more likely to consider African-American offenders to have higher risk scores than Caucasians, even though ethnicity information is not part of the input [220]. Similar situations have been observed in beauty contest judged by AI, which was biased against contestants with darker skin tones, or facial recognition AI that wrongly recognized Asian facial features [46].

The source of these biases often originate from data. For example, AI systems that have been trained on public datasets that contain mostly white persons, such as ImageNet, are likely to make errors when processing images of black persons. Similarly, in an application screening system, inappropriate features, such as the gender or race of applicants, might be unintentionally learned by the machine learning model [36, 37]. As a result, there is a need to unlearn these data, including the features and affected data items.

1.2 Challenges in Machine Unlearning

Before we can truly achieve machine unlearning, several challenges to removing specific parts of the training data need to be overcome. The challenges are summarized as follows.

Stochasticity of training. We do not know the impact of each data point seen during training on the machine learning model due to the stochastic nature of the training procedure [10]. Neural networks, for example, are usually trained on random mini-batches

[1] https://github.com/tamlhp/awesome-machine-unlearning

containing a certain number of data samples. Further, the order of the training batches is also random [10]. This stochasticity raises difficulties for machine unlearning as the specific data sample to be removed would need to be removed from all batches.

Incrementality of training. A model's training procedure is an incremental process [10]. In other words, the model update on a given data sample will affect the model performance on data samples fed into the model after this data. A model's performance on this given data sample is also affected by prior data samples. Determining a way to erase the effect of the to-be-removed training sample on further model performance is a challenge for machine unlearning.

Catastrophic unlearning. In general, an unlearned model usually performs worse than the model retrained on the remaining data [127, 128]. However, the degradation can be exponential when more data is unlearned. Such sudden degradation is often referred as catastrophic unlearning [127]. While several studies [38, 60] have explored ways to mitigate catastrophic unlearning by designing special loss functions, how to naturally prevent catastrophic unlearning is still an open question.

1.3 Contributions of this survey

The aim of this paper is to supply a complete examination of research studies on machine unlearning as well as a discussion on potential new research directions in machine unlearning. The contributions of our survey can therefore be summarized as follows.

- First, we show how to design an unlearning framework. We discuss the design requirements, different types of unlearning requests, and how to verify the unlearned model. The details can be found in §2.
- Second, we show how to define an unlearning problem in machine learning systems. This includes the formulation of exact unlearning and approximate unlearning as well as the definition of indistinguishability metrics to compare two given models (i.e., the unlearned model and the retrained model). The details are discussed in §3.
- Third, we discuss different scenarios of machine unlearning, including zero-glance unlearning, zero-shot unlearning, and few-shot unlearning. The details are provided in §4
- Fourth, we introduce a unified taxonomy that categorizes the machine unlearning approaches into three branches: model-agnostic methods, model-intrinsic methods, and data-driven methods. The details can be found in §5.
- Fifth, we compile a variety of regularly used datasets and open-source implementations to serve as a foundation for future machine unlearning research and benchmarking. The details are provided in §6.
- Finally, we highlight the findings, trends and the forthcoming according to our survey in §8. §9 then completes the paper.

1.4 Differences between this and previous surveys

Table 1 summarizes the differences between our survey and existing efforts to unify the field. It is noteworthy that machine unlearning is different from data deletion [54]. Some works focus on exploring theoretical foundations or subcategories of unlearning techniques [156, 205]. Both topics concern the right to be forgotten legislated and exercised across the world [115]. However, the latter focuses only on the data perspective following the General Data Protection Regulation (GDPR) [186], while machine unlearning also addresses privacy problems from a model perspective.

There are some other concepts that might be mistaken as machine unlearning, such as data redaction that aims to poison the label information of the data to be forgotten inside the model [45]. In other words, it forces the model make wrong predictions about the forgotten data. Although applicable in some setting, this approach is not fully compatible with machine unlearning as the forgotten data has to be known a priori when the original model is trained [45].

2 UNLEARNING FRAMEWORK

2.1 Unlearning Workflow

The unlearning framework in Fig. 1 presents the typical workflow of a machine learning model in the presence of a data removal request. In general, a model is trained on some data and is then used for inference. Upon a removal request, the data-to-be-forgotten is unlearned from the model. The unlearned model is then verified against privacy criteria, and, if these criteria are not met, the model is retrained, i.e., if the model still leaks some information about the forgotten data. There are two main components to this process: the *learning component* (left) and the *unlearning component* (right). The learning component involves the current data, a learning algorithm, and the current model. In the beginning, the initial model is trained from the whole dataset using the learning algorithm. The unlearning component involves an unlearning algorithm, the unlearned model, optimization requirements, evaluation metrics, and a verification mechanism. Upon a data removal request, the current model will be processed by an unlearning algorithm to forget the corresponding information of that data inside the model. The unlearning algorithm might take several requirements into account such as completeness, timeliness, and privacy guarantees. The outcome is an unlearned model, which will be evaluated against different performance metrics (e.g., accuracy, ZRF score, anamnesis index). However, to provide a privacy certificate for the unlearned model, a verification (or audit) is needed to prove that the model actually forgot the requested data and that there are no information leaks. This audit might include a feature injection test, a membership inference attack, forgetting measurements, etc.

If the unlearned model passes the verification, it becomes a new model for downstream tasks (e.g., inference, prediction, classification, recommendation). Otherwise, the remaining data, i.e., the original data excluding the data to be forgotten, needs to be used to retrain the model. Either way, the unlearning component will be called repeatedly upon a new removal request.

2.2 Unlearning Requests

Item Removal. Requests to remove certain items/samples from the training data are the most common requests in machine unlearning [10]. The techniques used to unlearn these data are described in detail in §5.

Feature Removal. In many scenarios, privacy leaks might not only originate from a single data item but also in a group of data

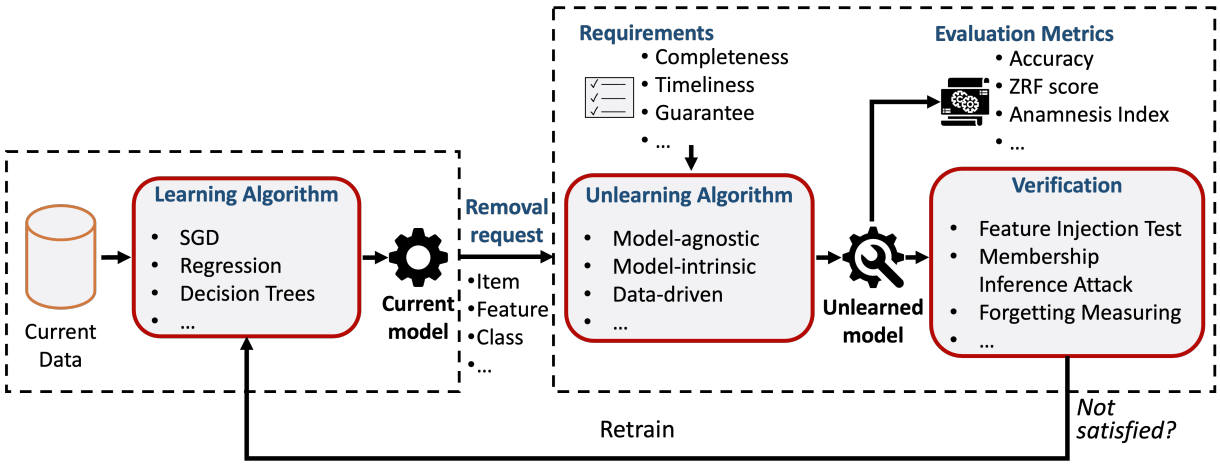

Figure 1: A Machine Unlearning Framework

with the similar features or labels [196]. For example, a poisoned spam filter might misclassify malicious addresses that are present in thousands of emails. Thus, unlearning suspicious emails might not enough. Similarly, in an application screening system, inappropriate features, such as the gender or race of applicants, might need to be unlearned for thousands of affected applications.

In such cases, naively unlearning the affected data items sequentially is imprudent as repeated retraining is computationally expensive. Moreover, unlearning too many data items can inherently reduce the performance of the model, regardless of the unlearning mechanism used. Thus, there is a need for unlearning data at the feature or label level with an arbitrary number of data items.

Warnecke et al. [196] proposed a technique for unlearning a group of training data based on influence functions. More precisely, the effect of training data on model parameter updates is estimated and formularized in closed-form. As a result of this formulation, influences of the learning sets act as a compact update instead of solving an optimisation problem iteratively (e.g., loss minimization). First-order and second-order derivatives are the keys to computing this update effectively [196].

Guo et al. [66] proposed another technique to unlearn a feature based on disentangled representation. The core idea is to learn the correlation between features from the latent space as well as the effects of each feature on the output space. Using this information, certain features can be progressively detached from the learnt model upon request, while the remaining features are still preserved to maintain good accuracy. However, this method is mostly applicable to deep neural networks in the image domain, in which the deeper convolutional layers become smaller and can therefore identify abstract features that match real-world data attributes.

Class Removal. There are many scenarios where the forgetting data belongs to single or multiple classes from a trained model. For example, in face recognition applications, each class is a person's face so there could potentially be thousands or millions of classes. However, when a user opts out of the system, their face information must be removed without using a sample of their face.

Similar to feature removal, class removal is more challenging than item removal because retraining solutions can incur many unlearning passes. Even though each pass might only come at a small computational cost due to data partitioning, the expense mounts up. However, partitioning data by class itself does not help the model's training in the first place, as learning the differences between classes is the core of many learning algorithms [170]. Although some of the above techniques for feature removal can be applied to class removal [196], it is not always the case as class information might be implicit in many scenarios.

Tarun et al. [172] proposed an unlearning method for class removal based on data augmentation. The basic concept is to introduce noise into the model such that the classification error is maximized for the target class(es). The model is updated by training on this noise without the need to access any samples of the target class(es). Since such impair step may disturb the model weights and degrade the classification performance for the remaining classes, a repair step is needed to train the model for one or a few more epochs on the remaining data. Their experiments show that the method can be efficient for large-scale multi-class problems (100 classes). Further, the method worked especially well with face recognition tasks because the deep neural networks were originally trained on triplet loss and negative samples so the difference between the classes was quite significant [118].

Baumhauer et al. [4] proposed an unlearning method for class removal based on a linear filtration operator that proportionally shifts the classification of the samples of the class to be forgotten to other classes. However, the approach is only applicable to class removal due to the characteristics of this operator.

Task Removal. Today, machine learning models are not only trained for a single task but also for multiple tasks. This paradigm, aka continual learning or lifelong learning [135], is motivated by the human brain, in which learning multiple tasks can benefit each other due to their correlations. This technique is also used overcome data sparsity or cold-start problems where there is not enough data to train a single task effectively.

However, in these settings too, there can be a need to remove private data related to a specific task. For example, consider a robot that is trained to assist a patient at home during their medical treatment. This robot may be asked to forget this assistance behaviour after the patient has recovered [101]. To this end, temporarily learning a task and forgetting it in the future has become a need for lifelong learning models.

In general, unlearning a task is uniquely challenging as continual learning might depend on the order of the learned tasks. Therefore, removing a task might create a catastrophic unlearning effect, where the overall performance of multiple tasks is degraded in a domino-effect [101]. Mitigating this problem requires the model to be aware of that the task may potentially be removed in future. Liu et al. [101] explains that this requires users to explicitly define which tasks will be learned permanently and which tasks will be learned only temporarily.

Stream Removal. Handling data streams where a huge amount of data arrives online requires some mechanisms to retain or ignore certain data while maintaining limited storage [169]. In the context of machine unlearning, however, handling data streams is more about dealing with a stream of removal requests.

Gupta et el. [67] proposed a streaming unlearning setting involving a sequence of data removal requests. This is motivated by the fact that many users can be involved in a machine learning system and decide to delete their data sequentially. Such is also the case when the training data has been poisoned in an adversarial attack and the data needs to be deleted gradually to recover the model's performance. These streaming requests can be either non-adaptive or adaptive. A non-adaptive request means that the removal sequence does not depend on the intermediate results of each unlearning request, whereas and adaptive request means that the data to be removed depends on the current unlearned model. In other words, after the poisonous data is detected, the model is unlearned gradually so as to decide which data item is most beneficial to unlearn next.

2.3 Design Requirements

Completeness (Consistency). A good unlearning algorithm should be complete [12], i.e. the unlearned model and the retrained model make the same predictions about any possible data sample (whether right or wrong). One way to measure this consistency is to compute the percentage of the same prediction results on a test data. This requirement can be designed as an optimization objective in an unlearning definition (§3.2) by formulating the difference between the output space of the two models. Many works on adversarial attacks can help with this formulation [23, 167].

Timeliness. In general, retraining can fully solve any unlearning problem. However, retraining is time-consuming, especially when the distribution of the data to be forgotten is unknown [10, 12]. As a result, there needs to be a trade-off between completeness and timeliness. Unlearning techniques that do not use retraining might be inherently not complete, i.e., they may lead to some privacy leaks, even though some provable guarantees are provided for special cases [64, 116, 126]. To measure timeliness, we can measure the speed up of unlearning over retraining after an unlearning request is invoked.

It is also worth recognizing the cause of this trade-off between retraining and unlearning. When there is not much data to be forgotten, unlearning is generally more beneficial as the effects on model accuracy are small. However, when there is much forgetting data, retraining might be better as unlearning many times, even bounded, may catastrophically degrade the model's accuracy [12].

Accuracy. An unlearned model should be able to predict test samples correctly. Or at least its accuracy should be comparable to the retrained model. However, as retraining is computationally costly, retrained models are not always available for comparison. To address this issue, the accuracy of the unlearned model is often measured on a new test set, or it is compared with that of the original model before unlearning [72].

Light-weight. To prepare for unlearning process, many techniques need to store model checkpoints, historical model updates, training data, and other temporary data [10, 72, 102]. A good unlearning algorithm should be light-weight and scale with big data. Any other computational overhead beside unlearning time and storage cost should be reduced as well [10].

Provable guarantees. With the exception of retraining, any unlearning process might be inherently approximate. It is practical for an unlearning method to provide a provable guarantee on the unlearned model. To this end, many works have designed unlearning techniques with bounded approximations on retraining [64, 116, 126]. Nonetheless, these approaches are founded on the premise that models with comparable parameters will have comparable accuracy.

Model-agnostic. An unlearning process should be generic for different learning algorithms and machine learning models [10], especially with provable guarantees as well. However, as machine learning models are different and have different learning algorithms as well, designing a model-agnostic unlearning framework could be challenging.

Verifiability. Beyond unlearning requests, another demand by users is to verify that the unlearned model now protects their privacy. To this end, a good unlearning framework should provide end-users with a verification mechanism. For example, backdoor attacks can be used to verify unlearning by injecting backdoor samples into the training data [166]. If the backdoor can be detected in the original model while not detected in the unlearned model, then verification is considered to be a success. However, such verification might be too intrusive for a trustworthy machine learning system and the verification might still introduce false positive due to the inherent uncertainty in backdoor detection.

2.4 Unlearning Verification

The goal of unlearning verification methods is to certify that one cannot easily distinguish between the unlearned models and their retrained counterparts [174]. While the evaluation metrics (§6.3) are theoretical criteria for machine unlearning, unlearning verification can act as a certificate for an unlearned model. They also include best practices for validating the unlearned models efficiently.

It is noteworthy that while unlearning metrics (in §3.1) and verification metrics share some overlaps, the big difference is that the former can be used for optimization or to provide a bounded guarantee, while the latter is used for evaluation only.

Feature Injection Test. The goal of this test is to verify whether the unlearned model has adjusted the weights corresponding to the removed data samples based on data features/attributes [79]. The idea is that if the set of data to be forgotten has a very distinct feature distinguishing it from the remaining set, it gives a strong signal for the model weights. However, this feature needs to be correlated with the labels of the set to be forgotten, otherwise the model might not learn anything from this feature.

More precisely, an extra feature is added for each data item such that it is equal to zero for the remaining set and is perfectly correlated with the labels of the set to forget. Izzo et al. [79] applied this idea with linear classifiers, where the weight associated with this extra feature is expected to be significantly different from zero after training. After the model is unlearned, this weight is expected to become zero. As a result, the difference of this weight can be plotted before and after unlearning as a measure of effectiveness of the unlearning process.

One limitation of this verification method is that the current solution is only applicable for linear and logistic models [79]. This is because these models have explicit weights associated with the injected feature, whereas, for other models such as deep learning, injecting such a feature as a strong signal is non-trivial, even though the set to be forgotten is small. Another limitation to these types of methods is that an injected version of the data needs to be created so that the model can be learned (either from scratch or incrementally depending on the type of the model).

Forgetting Measuring. Even after the data to be forgotten has been unlearned from the model, it is still possible for the model to carry detectable traces of those samples [80]. Jagielski et al. [80] proposed a formal way to measure the forgetfulness of a model via privacy attacks. More precisely, a model is said to α-forget a training sample if a privacy attack (e.g., a membership inference) on that sample achieves no greater than success rate α. This definition is more flexible than differential privacy because a training algorithm is differentially private only if it immediately forgets every sample it learns. As a result, this definition allows a sample to be temporarily learned, and measures how long until it is forgotten by the model.

Information Leakage. Many machine learning models inherently leak information during the model updating process [23]. Recent works have exploited this phenomenon by comparing the model before and after unlearning to measure the information leakage. More precisely, Salem et al. [148] proposed an adversary attack in the image domain that could reconstruct a removed sample when a classifier is unlearned on a data sample. Brockschmidt et al. [211] suggested a similar approach for the text domain. Chen et al. [23] introduced a membership inference attack to detect whether a removed sample belongs to the learning set. Compared to previous works [149, 161], their approach additionally makes use of the posterior output distribution of the original model, besides that of the unlearned model. Chen et al. [23] also proposed two leakage metrics, namely the degradation count and the degradation rate.

- The *degradation count:* is defined as the ratio between the number of target samples whose membership can be inferred by the proposed attack with higher confidence compared to traditional attacks and the total number of samples.
- The *degradation rate:* is defined the average improvement rate of the confidence of the proposed attack compared to traditional attacks.

Membership Inference Attacks. This kind of attack is designed to detect whether a target model leaks data [23, 161, 175]. Specifically, an inference model is trained to recognise new data samples from the training data used to optimize the target model. In [161], a set of shallow models were trained on a new set of data items different from the one that the target model was trained on. The attack model was then trained to predict whether a data item belonged to the training data based on the predictions made by shallow models for training as well as testing data. The training set for the shallow and attack models share similar data distribution to the target model. Membership inference attacks are helpful for detecting data leaks. Hence, they are useful for verifying the effectiveness of the machine unlearning [23].

Backdoor attacks. Backdoor attacks were proposed to inject backdoors to the data for deceiving a machine learning model [187]. The deceived model makes correct predictions with clean data, but with poison data in a target class as a backdoor trigger, it makes incorrect predictions. Backdoor attacks were used to verify the effectiveness of machine unlearning in [166, 167]. Specifically, the setting begins with training a model that has a mixture of clean and poison data items across all users. Some of the users want their data deleted. If the users' data are not successfully deleted, the poison samples will be predicted as the target class. Otherwise, the model will not predict the poison samples as the target class. However, there is no absolute guarantee that this rule is always correct, although one can increase the number of poison samples to make this rule less likely to fail.

Slow-down attacks. Some studies focus on the theoretical guarantee of indistinguishability between an unlearned and a retrained models. However, the practical bounds on computation costs are largely neglected in these papers [116]. As a result, a new threat has been introduced to machine unlearning where poisoning attacks are used to slow down the unlearning process. Formally, let $h_0 = A(D)$ be an initial model trained by a learning algorithm A on a dataset D. The goal of the attacker is to poison a subset $D_{poison} \subset D$ such as to maximize the computation cost of removing D_{poison} from $\hat{h}$ using an unlearning algorithm U. Marchant et al. [116] defined and estimated an efficient computation cost for certifying removal methods. However, generalizing this computation cost for different unlearning methods is still an open research direction.

Interclass Confusion Test. The idea of this test is to investigate whether information from the data to be forgotten can still be inferred from an unlearned model [58]. Different from traditional approximate unlearning definitions that focus on the indistinguishability between unlearned and retrained models in the parameter space, this test focuses on the output space. More precisely, the test involves randomly selecting a set of samples $S \subset D$ from two chosen classes in the training data D and then randomly swapping the label assignment between the samples of different classes to result in a confused set S'. Together S' and $D \setminus S$ form a new training dataset D', resulting in a new trained model. S' is considered to be the forgotten data. From this, Goet et al. [58] computes a forgetting

score from a confusion matrix generated by the unlearned model. A lower forgetting score means a better unlearned model.

Federated verification. Unlearning verification in federated learning is uniquely challenging. First, the participation of one or a few clients in the federation may subtly change the global model's performance, making verification in the output space challenging. Second, verification using adversarial attacks is not applicable in the federated setting because it might introduce new security threats to the infrastructure [53]. As a result, Gao et al. [53] proposes a verification mechanism that uses a few communication rounds for clients to verify their data in the global model. This approach is compatible with federated settings because the model is trained in the same way where the clients communicate with the server over several rounds.

Cryptographic proofs. Since most of existing verification frameworks do not provide any theoretical guarantee, Eisenhofer et al. [44] proposed a cryptography-informed protocol to compute two proofs, i.e. proof of update (the model was trained on a particular dataset D) and proof of unlearning (the forget item d is not a member of D). The core idea of the proof of update is using SNARK [7] data structure to commit a hash whenever the model is updated (learned or unlearned) while ensuring that: (i) the model was obtained from the remaining data, (ii) the remaining data does not contain any forget items, (iii) the previous forget set is a subset of the current forget set, and (iv) the forget items are never re-added into the training data. The core idea of the proof of unlearning is using the Merkle tree to maintain the order of data items in the training data so that an unlearned item cannot be added to the training data again. While the approach is demonstrated on SISA (efficient retraining) [10], it is applicable for any unlearning method.

3 UNLEARNING DEFINITION

3.1 Problem Formulation

While the application of machine unlearning can originate from security, usability, fidelity, and privacy reasons, it is often formulated as a privacy preserving problem where users can ask for the removal of their data from computer systems and machine learning models [10, 54, 57, 154]. The forgetting request can be motivated by security and usability reasons as well. For example, the models can be attacked by adversarial data and produce wrong outputs. Once these types of attacks are detected, the corresponding adversarial data has to be removed as well without harming the model's predictive performance.

When fulfilling a removal request, the computer system needs to remove all user's data and 'forget' any influence on the models that were trained on those data. As removing data from a database is considered trivial, the literature mostly concerns how to unlearn data from a model [64, 79, 126, 182].

To properly formulate an unlearning problem, we need to introduce a few concepts. First, let us denote $\mathcal{Z}$ as an example space, i.e., a space of data items or examples (called samples). Then, the set of all possible training datasets is denoted as $\mathcal{Z}^*$. One can argue that $\mathcal{Z}^* = 2^{\mathcal{Z}}$ but that is not important, as a particular training dataset $D \in Z^*$ is often given as input. Given D, we want to get a machine learning model from a hypothesis space $\mathcal{H}$. In general, the hypothesis space $\mathcal{H}$ covers the parameters and the meta-data of the models. Sometimes, it is modeled as $\mathcal{W} \times \Theta$, where $\mathcal{W}$ is the parameter space and Θ is the metadata/state space. The process of training a model on D in the given computer system is enabled by a learning algorithm, denoted by a function $A : \mathcal{Z}^* \to \mathcal{H}$, with the trained model denoted as $A(D)$.

Table 2: Important notations

Symbols	Definition
$\mathcal{Z}$	example space
D	the training dataset
D_f	forgetting set (the data to be forgotten)
$D_r = D \setminus D_f$	retained set (the remaining data)
$A(.)$	a learning algorithm
$U(.)$	an unlearning algorithm
$\mathcal{H}$	hypothesis space of models
$w = A(D)$	Parameters of the model trained on D by A
$w_r = A(D_r)$	Parameters of the model trained on D_r by A
$w_u = U(.)$	Parameters of the model unlearned by $U(.)$

To support forgetting requests, the computer system needs to have an unlearning mechanism, denoted by a function U, that takes as input a training dataset $D \in Z^*$, a forget set $D_f \subset D$ (data to forget) and a model $A(D)$. It returns a sanitized (or unlearned) model $U(D, D_f, A(D)) \in \mathcal{H}$. The unlearned model is expected to be the same or similar to a retrained model $A(D \setminus D_f)$ (i.e., a model as if it had been trained on the remaining data). Note that A and U are assumed to be randomized algorithms, i.e., the output is non-deterministic and can be modelled as a conditional probability distribution over the hypothesis space given the input data [116]. This assumption is reasonable as many learning algorithms are inherently stochastic (e.g., SGD) and some floating-point operations involve randomness in computer implementations [10]. Another note is that we do not define the function U precisely before-hand as its definition varies with different settings.

Table 2 summarizes important notations.

3.2 Exact Unlearning (Perfect Unlearning)

The core problem of machine unlearning involves the comparison between two distributions of machine learning models [10, 11, 173]. Let $Pr(A(D))$ define the distribution of all models trained on a dataset D by a learning algorithm $A(.)$. Let $Pr(U(D, D_f, A(D)))$ be the distribution of unlearned models. The reason why the output of $U(.)$ is modelled as a distribution rather than a single point is that learning algorithms $A(.)$ and unlearning algorithms $U(.)$ are randomized as mentioned above.

Definition 1 (Exact unlearning - special case). *Given a learning algorithm $A(.)$, a dataset D, and a forget set $D_f \subseteq D$, we say the process $U(.)$ is an exact unlearning process iff:*

$$Pr(A(D \setminus D_f)) = Pr(U(D, D_f, A(D))) \quad (1)$$

Two key aspects can be drawn from this definition. First, the definition does not require that the model $A(D)$ be retrained from scratch on $D \setminus D_f$. Rather, it requires some evidence that it is likely to be a model that is trained from scratch on $D \setminus D_f$. Second, two models trained with the same dataset should belong to the same distribution. However, defining this distribution is tricky. So to avoid the unlearning algorithm being specific to a particular training dataset, we have a more general definition [11, 57]:

Definition 2 (Exact unlearning - general case). *Given a learning algorithm $A(.)$, we say the process $U(.)$ is an exact unlearning process iff $\forall \mathcal{T} \subseteq \mathcal{H}, D \in Z^*, D_f \subset D$:*

$$Pr(A(D \setminus D_f) \in \mathcal{T}) = Pr(U(D, D_f, A(D)) \in \mathcal{T}) \quad (2)$$

This definition allows us to define a metric space the models belong to (and consequently for the distributions). A model can be viewed either as just a mapping of inputs to outputs in which case $Pr(.)$ are distributions over a function space (i.e., continuous function with the supremum metric), or as the specific parameters $\boldsymbol{\theta}$ for a model architecture, in which case $Pr(.)$ are distributions over the weight space (e.g., some finite dimensional real vector space with the Euclidean norm). This ambiguity leads to two notions of exact unlearning:

- *Distribution of weights*: Eq. 2 implies the zero difference in the distribution of weights, i.e., $Pr(w_r) = Pr(w_u)$, where the parameters of models w_r learned by $A(D_r)$ and w_u are the parameters of the models given by $U(.)$.
- *Distribution of outputs:* Eq. 2 implies zero difference in the distribution of outputs, i.e., $Pr(M(X; w_r)) = Pr(M(X; w_u))$, $\forall X \subseteq \mathcal{Z}$, where $M(.)$ is the parameterized mapping function from the input space $\mathcal{Z}$ to the output space (i.e., the machine learning model). This definition is sometimes referred to as *weak unlearning* [4].

If the unlearning mechanism $U(.)$ is implemented as retraining itself, equality is absolutely guaranteed. For this reason, retraining is sometimes considered to be the only exact unlearning method. However, retraining inherently involves high computation costs, especially for large models [173]. Another disadvantage of retraining is that it cannot deal with batch settings, where multiple removal requests happen simultaneously or are grouped in a batch.

There are many different metrics for comparing numerical distributions over the output space and the weight space. However, doing so is expensive (e.g., generating a sample in these distributions involves training the whole model). To mitigate this issue, some approaches design an alternative metric on a point basis to compute the distance between two models, either in the output space or in the weight space [161].

3.3 Approximate Unlearning (Bounded/Certified Unlearning)

Approximate unlearning approaches attempt to address these cost-related constraints. In lieu of retraining, these strategies: perform computationally less costly actions on the final weights [63, 64, 154]; modify the architecture [4]; or filter the outputs [4]. Approximate unlearning relaxes Def. 2 as follows [64].

Definition 1 (ϵ-Approximate Unlearning). Given $\epsilon > 0$, an unlearning mechanism U performs ϵ-certified removal for a learning algorithm A if $\forall \mathcal{T} \subseteq \mathcal{H}, D \in Z^*, z \in D$:

$$e^{-\epsilon} \leq \frac{Pr(U(D, z, A(D)) \in \mathcal{T})}{Pr(A(D \setminus z) \in \mathcal{T})} \leq e^{\epsilon} \quad (3)$$

where z is the removed sample.

It is noteworthy that Eq. 3 defines the bounds on a single sample z only. It is still an open question as to whether constant bounds can be provided for bigger subsets of D. Moreover, the reason why we have the $[e^{-\epsilon}, e^{\epsilon}]$ bounds is that the probability distributions are often modeled by log functions, in which Eq. 3 is equivalent to:

$$-\epsilon \leq \log\left[Pr(U(D, z, A(D)) \in \mathcal{T}) - Pr(A(D \setminus z) \in \mathcal{T})\right] \leq \epsilon \quad (4)$$

or:

$$\log ||Pr(U(D, z, A(D)) \in \mathcal{T}) - Pr(A(D \setminus z) \in \mathcal{T})|| \leq \epsilon \quad (5)$$

where $||.||$ is an absolute distance metric on the weight space or the output space. A relaxed version of ϵ-approximate unlearning is also defined in [126]:

Definition 3 ((ϵ,δ)-Approximate Unlearning). *Given $\epsilon, \delta > 0$, an unlearning mechanism U performs ϵ-certified removal for a learning algorithm A if $\forall \mathcal{T} \subseteq \mathcal{H}, D \in Z^*, z \in D$:*

$$Pr(U(D, z, A(D)) \in \mathcal{T}) \leq e^{\epsilon} Pr(A(D \setminus z) \in \mathcal{T}) + \delta \quad (6)$$

and

$$Pr(A(D \setminus z) \in \mathcal{T}) \leq e^{\epsilon} Pr(U(D, z, A(D)) \in \mathcal{T}) + \delta \quad (7)$$

In other words, δ upper bounds the probability for the max-divergence bound in Eq. 3 to fail.

Relationship to differential privacy. Differential privacy states that:

$$\forall \mathcal{T} \subseteq \mathcal{H}, D, D' : e^{-\epsilon} \leq \frac{Pr(A(D) \in \mathcal{T})}{Pr(A(D \setminus z) \in \mathcal{T})} \leq e^{\epsilon} \quad (8)$$

where z is the removed sample. Differential privacy implies approximate unlearning: deleting the training data is not a concern if algorithm A never memorises it in the first place [64]. However, this is exactly the contradiction between differential privacy and machine unlearning. If A is differentially private for any data, then it does not learn anything from the data itself [10]. In other words, differential privacy is a very strong condition, and most differentially private models suffer a significant loss in accuracy even for large ϵ [1, 17].

3.4 Indistinguishability Metrics

To compare the two models in Def. 2, we need to define a distance metric $d(.)$ between $Pr(A(D \setminus D_f) \in \mathcal{T})$ and $Pr(U(D, D_f, A(D)) \in \mathcal{T})$ ($\forall \mathcal{T} \subseteq \mathcal{H}$) in either the weight (parameter) space or the output space. To this end, several distance metrics have been studied:

ℓ_2-distance. Wu et al. [201] proposed using a Euclidean norm to compare the weights of $A(D_r)$ and the weights of $U(D, D_f, A(D))$. This is also termed as *verification error* [201]. Despite being simple, this metric has several limitations: (1) It is costly to compute this verification error as we need to also calculate $A(D_r)$ (through naive retraining). If the computational cost is cheap, machine unlearning is not necessary in the first place. (2) It is possible for two models having same training set and initialisation to have different weights [81] due to training stochasticity and uncertainties in floating-point operations. Therefore, it is quite tricky to define a threshold for this error.

KL Divergence. The Kullback-Leiber (KL) divergence or the Jensen-Shannon divergence is a popular distance metric between two distributions. Golatkar et al. [60] considered this divergence to measure the distance between two models in the parameter space. Although it might not require computing $A(D_r)$, it is necessary to have final distributions of models train on D_r for computing the divergence. Distribution modeling is non-trivial and might involve sampling of many models trained on D_r as well.

Weight Leakage. Some studies measure privacy leaks of the removed sample z from the parameters of the unlearned model [43, 64, 154]. These works assume that a model's weight distribution does not leak information about z if it was not trained on z. However, measuring this privacy leakage is non-trivial and only well-defined for special classes of models. Guo et al. [64] proposed such a metric for linear classifiers via gradients. More precisely, a model w^* is trained on D if the gradient $\nabla L(D; w^*) = 0$, where $L(.)$ is an empirical risk (loss) of a linear model. This is because $\arg\min_w L(D; w)$ is uniquely determined for such $L(.)$. As a result, if the gradient $\nabla L(D \setminus z; w_u)$, where w_u is the parameters of the model returned by U, is non-zero then the model either does not finish training or is not trained on $D \setminus z$. The former case can be safely ignored as we are not interested in non-converged models. However, the latter case indirectly implies that the model was trained on a dataset that included z, and thus it reveals some information about z. As a result, the gradient $\nabla L(D \setminus z; w_u)$ becomes an objective function for minimizing (or providing a bound) in those works [64, 154]. Although this metric does provide an efficient way to verify the unlearning result, the above assumptions are not always correct from a numerical perspective, especially for non-linear models [67]. Using these metrics also requires access to the remaining data as well as the loss function, which are not always available.

4 OTHER UNLEARNING SCENARIOS

Beside the exact unlearning and approximate unlearning settings, there are other settings where access to the data, whether they are to be forgotten or retained, is restricted for security reasons.

4.1 Zero-glance Unlearning

Traditional privacy settings in machine unlearning assume there is access to all the training data before forgetting. Tarun et al. [172] proposed to work in a stricter setting where once the user had made a request to forgetting their data (for example, their face in a facial recognition model), the organization could not use those samples even for the purposes of model weight manipulation. Formally, the zero-glance setting means that the unlearning algorithm uses only the retained data:

$$w_u = U(D_{r\_sub}, A(D)) \tag{9}$$

where w_u is the unlearned model and $D_{r\_sub} \subseteq D_r$ is a subset of samples drawn from the retained dataset $D_r = D \setminus D_f$. The smaller the D_{r_sub}, the better the privacy.

Error-maximizing noise. Unlearning without knowing the data to be forgotten is non-trivial. Tarun et al. [172] relaxed the setting somewhat by defining a set of classes to be forgotten, which should be completely removed from the organization. More precisely, given a sample x and class label $y \in C$, and a set of classes $C = \{1, \ldots, K\}$, the training data D is now a set of pairs (x, y). Let C_f denote the set of classes that needs to be forgotten by the model, and let D_f be the data collection the classes to be forgotten belong to. The unlearned model is computed as: $w_u = U(D_{r\_sub}, C_f, A(D))$.

Tarun et al. [172] proposed learning a noise matrix for the classes to be forgotten C_f by maximizing the model's loss. In other words, a set of noise samples $\mathcal{N}$ are generated from $C_f, A(D)$ to approximate the data to be forgotten D_f. Then, the model $A(D)$ is trained for 1 epoch on $D_{r\_sub} \cup \mathcal{N}$ to damage the model parameters on the forgotten classes, thus inducing unlearning. After that, the impaired model is trained for 1 epoch on D_{r_sub} so as to repair any damage on the retained classes. Sometimes, it takes more epochs or a larger D_{r_sub} to enhance the unlearned model's performance. This is also a way to trade the advantages between retraining and unlearning (accuracy vs. time), which is not available in naive retraining.

In simple terms, existing methods for zero-glance unlearning try to transform the problem into its original form, where $w_u = U(D, D_f, A(D))$ by generating an approximate version of D_f.

4.2 Zero-shot Unlearning

If the training data is not accessible by an unlearning method U, the scenario becomes zero-shot unlearning. That is, the unlearning objective in Def. 2 becomes:

$$Pr(A(D \setminus D_f) \in \mathcal{T}) \approx Pr(U(D_f, A(D)) \in \mathcal{T}) \tag{10}$$

However, it is non-trivial to solve this problem in generic cases. Similar to the zero-glance unlearning, Chundawat et al. [29] studied such a setting for classification with classes to be forgotten with the idea being that the outputs of the unlearned model should resemble those of a retrained model. That is, the unlearning objective they want to achieve was: for some $X \subset \mathcal{Z}$:

$$M(X; w_u) \approx M(X; w_r) \tag{11}$$

where $w_r = A(D \setminus D_f)$ and $w_u = U(C_f, A(D))$. Note that the data to be forgotten D_f is not accessible to the unlearning algorithm $U(.)$. But, rather, as C is a generally available information, $U(.)$ has access to the retained classes $C_r = C \setminus C_f$.

Error-minimization noise. Chundawat et al. [29] reused error-maximizing noise of [172] to generate an impaired version of D_f such that the model parameters of $A(D)$ are damaged when being trained on those noise samples. To handle the lack of access to retain data, Chundawat et al. [29] proposed error-minimizing noise to generate an approximate version of D_r such that the impaired model could be trained to mitigate the performance degradation.

Gated knowledge transfer. Unfortunately, the above error max-min approach yields poor unlearning outcomes as the generated noise is somewhat adhoc. Hence, inspired by [124], Chundawat et al. [29] proposed a knowledge distillation mechanism with a crucial trick: a special type of 'gate' is introduced to deny any knowledge of the classes to be forgotten C_f so as to prevent a teacher model from passing knowledge to a student model.

4.3 Few-shot Unlearning

Few-shot unlearning is actually more similar to zero-glance setting rather than the zero-shot setting in that the unlearning algorithm only receives a small portion of the data to be forgotten D_f. More specifically, the unlearned model is computed as:

$$w_u = U(D, D_{f\_sub}, A(D)) \tag{12}$$

where $D_{f\_sub} \subseteq D_f$. This setting is useful for cases where the set to be forgotten D_f contains mislabeled data or one wants to remove some data's malicious effects on a model. However, access to D_f can be undermined [207] due to privacy regulations. Also, in most cases, D_{f_sub} is much smaller than D_f, i.e., $D_{f\_sub} \ll |D_f|$.

Model inversion. Yoon et al. [207] introduced a framework for few-shot unlearning on the basis of model inversion. First, a proxy for the learning set is retrieved from the given model via a model

inversion mechanism. Second, a filtration method eliminates some of the retrieved data that may be responsible for the undesirable behaviour while interpolating the few target samples. Finally, the filtered data is used within a relearning process.

The proposed framework even works on stricter setting where the unlearning process can not access the original training data, i.e., $w_u = U(D_{f\_sub}, A(D))$ [207] . However, it is only applicable to classification models with a cross-entropy loss.

Influence approximation. Peste et al. [138] proposed an unlearning process based on influence functions. The existing efficient influence-based unlearning process needs the calculation and the inverse of the Hessian matrix of loss from the model to determine the impact of any sample. However, those mentioned operations on the Hessian matrix are extremely costly, especially for high-dimensional models. Peste et al. [138] approximated it by using an empirical Fisher Information Matrix (FIM), which uses rank-one updates to allow easy computation and fast matrix inversion. As only the to-be-removed data is accessible and erasing the samples is simple via closed-form updates, those approximations gain a significant practical advantage.

5 UNLEARNING ALGORITHMS

As mentioned in the Section 1, machine unlearning can remove data and data linkages without retraining the machine learning model from scratch, saving time and computational resources [21, 191]. The specific approaches of machine unlearning can be categorized into model-agnostic, model-intrinsic, and data-driven approaches.

5.1 Model-Agnostic Approaches

Model-agnostic machine unlearning methodologies include unlearning processes or frameworks that are applicable to different models. However, in some cases, theoretical guarantees are only provided for a class of models (e.g., linear models). Nonetheless, they are still considered to be model-agnostic as their core ideas are applicable to complex models (e.g. deep neural networks) with practical results.

Differential Privacy. Differential privacy was first proposed to bound a data sample's influence on a machine learning model [42]. ϵ-differential privacy unlearns a data sample by setting $\epsilon = 0$, where ϵ bounds the level of change in any model parameters affected by that data sample [10, 173]. However, Bourtoule et al. [10] notes that the algorithm cannot learn from the training data in such a case. Gupta et el. [67] proposed a differentially private unlearning mechanism for streaming data removal requests. These requests are adaptive as well, meaning the data to be removed depends on the current unlearned model. The idea, which is based on differential privacy, can be roughly formulated as:

$$\Pr(U(D, s, A(D)) \in \mathcal{T}) \leq e^{\epsilon} Pr(A(D \setminus s) \in \mathcal{T}) + \beta \qquad (13)$$

for all adaptive removal sequences $s = (z_1, \ldots, z_k)$. One weakness of this condition is that it only guarantees the upper bound of the unlearning scheme compared to full retraining. However, its strength is that it supports a user's belief that the system has engaged in full retraining. Finally, an unlearning process is developed by a notion of differentially private publishing functions and a theoretical reduction from adaptive to non-adaptive sequences. Differentially private publishing functions guarantee that the model before and after an unlearning request do not differ too much [47].

Certified Removal Mechanisms. Unlearning algorithms falling into this category are the ones following the original approximate definition of machine unlearning [60, 64]. While Guo et al. [64] focus on theoretical guarantees for linear models and convex losses, Golatkar et al. [60] introduce a computable upper bound for SGD-based learning algorithms, especially deep neural networks. The core idea is based on the notion of perturbation (noise) to mask the small residue incurred by the gradient-based update (e.g., a one-step Newton update [91]). The idea is applicable to other cases, although no theoretical guarantees are provided [10].

More precisely, certified removal mechanisms mainly accommodate those linear models that minimize a standardized empirical risk, which is the total value of a convex loss function that measures the distance of the actual value from the expected one [116]. However, one has to rely on a customized learning algorithm that optimizes a perturbed version of the regularized empirical risk, where the added noise is drawn from a standard normal distribution. This normalized noise allows conventional convex optimization techniques to solve the learning problem with perturbation. As a result, the unlearning request can be done by computing the model perturbation towards the regularized empirical risk on the remaining data. The final trick is that this perturbation can be approximated by the influence function [91], which is computed by inverting the Hessian on training data and the gradient of the data to be forgotten [116]. However, the error of model parameters in such a computation can be so large that the added noise cannot mask it. Therefore, if the provided theoretical upper bound exceeds a certain threshold, the unlearning algorithm resorts to retraining from scratch [116].

Following this idea, Neel et al. [126] provided further extensions, namely regularized perturbed gradient descent and distributed perturbed gradient descent, to support weak convex losses and provide theoretical guarantees on indistinguishability, accuracy, and unlearning times.

Ullah et al. [182] continued studying machine unlearning in the context of SGD and streaming removal requests. They define the notation of total variation stability for a learning algorithm:

$$\sup_{D, D' : |D \setminus D'| + |D' \setminus D|} \Delta(A(D), A(D')) \leq \rho \qquad (14)$$

where $\Delta(.)$ is the largest possible difference between the two probabilities such that they can assign to the same event, aka total variance distance [184]. This is also a special case of the optimal transportation cost between two probability distributions [95]. In other words, a learning algorithm $A(.)$ is said to be ρ-TV-stable if given any two training datasets D and D', as long as they have 1 common data item, the cost of transporting from the model distribution $A(D)$ to $A(D')$ is bounded by ρ. For any $1/n \leq \rho < \infty$, Ullah et al. [182] proved that there exists an unlearning process that satisfies exact unlearning at any time in the streaming removal request while the model accuracy and the unlearning time are bounded w.r.t. ρ.

Statistical Query Learning. Statistical query learning is a form of machine learning that trains models by querying statistics on the training data rather than itself [12]. In this form, a data sample can be forgotten efficiently by recomputing the statistics over the

Table 3: Comparison of unlearning methods

Unlearning Methods	Unlearning Scenarios					Design Requirements						Unlearning Requests				
	Exact	Approximate	Zero-glance	Zero-shot	Few-shot	Completeness	Timeliness	Accuracy	Lightweight	Guarantees	Verifiability	Item	Feature	Class	Task	Stream
Model-agnostic																
Differential privacy [47, 67]	✗	✓	–	–	–	✓	✓	–	✓	✓	–	✓	✗	✗	✗	✓
Certified removal [60, 64, 126, 182]	–	✓	✗	✗	–	–	✓	✓	✓	✓	–	✓	✗	✗	✗	✓
Statistical query learning [12]	–	✓	✗	–	✗	✓	✓	–	✓	–	–	✓	✗	✗	✗	–
Decremental learning [25, 57]	✗	–	✗	–	–	✗	✓	✓	–	–	–	✓	✗	✗	✗	✗
Knowledge adaptation [28, 89, 192, 215]	✗	✓	–	–	–	–	–	–	✗	✗	–	✓	–	–	–	–
Parameter sampling [128]	✗	✓	✓	✗	–	–	–	–	–	–	–	✓	✗	✗	✗	✗
Model-intrinsic																
Softmax classifiers [4]	✗	✓	✓	✗	–	✓	–	–	✓	✗	✓	✗	✗	✓	✗	✗
Linear models [79, 99]	✓	✗	✗	–	✗	–	✓	–	✓	✓	✓	✓	✗	✗	✗	✗
Tree-based models [153, 203]	✗	–	✗	–	✗	✗	–	✓	✗	✗	✗	✓	✗	✗	✗	✗
Bayesian models [127]	–	✓	✗	✗	✗	–	–	✓	–	✓	–	✓	✗	✗	✗	✗
DNN-based models [3, 59, 61, 62, 72, 121, 214]	✗	✓	✗	✗	✗	-	-	✓	-	-	-	✓	✗	✗	✗	✗
Data-driven																
Data partition [2, 10, 41]	✓	✗	✓	✗	✓	✓	✗	–	✗	✓	✓	✓	✗	–	✗	–
Data augmentation [75, 157, 172, 208]	✗	✓	✗	✗	✗	–	–	–	–	✗	–	–	✗	✓	✗	✗
Data influence [14, 138, 212]	✗	✓	✗	✓	–	–	✓	–	✓	✓	–	✓	✗	✗	✗	✗

✓: fully support ✗: no support –: partially or indirectly support []: representative citations

remaining data [10]. More precisely, statistical query learning assumes that most of the learning algorithms can be represented as a sum of some efficiently computable transformations, called statistical queries [87]. These statistical queries are basically requests to an oracle (e.g., a ground truth) to estimate a statistical function over all training data. Cao et al. [12] showed that this formulation can generalize many algorithms for machine learning, such as the Chi-square test, naive Bayes, and linear regression. For example, in naive Bayes, these statistical queries are indicator functions that return 1 when the output is a target label and zero otherwise [12]. In the unlearning process, these queries are simply recomputed over the remaining data. The approach is efficient as these statistical functions are computationally efficient in the first place. Moreover, statistical query learning also supports adaptive statistical queries, which are computed based on the prior state of the learning models, including k-means, SVM, and gradient descent. Although this time, the unlearning update makes the model not convergent any more, only a few learning iterations (adaptive statistical queries) are needed since the model starts from an almost-converged state. Moreover, if the old results of the summations are cached, say, via dynamic programming, then the speedup might be even higher.

The limitation of this approach is that it does not scale with complex models such as deep neural networks. Indeed, in complex models, the number of statistical queries could become exponentially large [10], making both the unlearning and relearning steps less efficient.

In general, statistical query learning supports item removal and can be partially applied to stream removal [67] as well, although the streaming updates to the summations could be unbounded. It supports exact unlearning, but only partially when the statistical queries are non-adaptive. It also partially supports zero-shot unlearning, because only the statistics over the data need to be accessed, not the individual training data items.

Decremental Learning. Decremental learning algorithms were originally designed to remove redundant samples and reduce the training load on the processor for support vector machines (SVM) [15, 25, 39, 146, 180, 181] and linear classification [84, 85, 179]. As such, they focus on accuracy rather than the completeness of the machine unlearning.

Ginart et al. [57] developed decremental learning solutions for k-means clustering based on quantization and data partition. The idea of quantization is to ensure that small changes in the data do not change the model. Quantization helps to avoid unnecessary unlearning so that accuracy is not catastrophically degraded. However, it is only applicable when there are few model parameters compared to the size of the dataset. The idea behind the data partitioning is to restrict the data's influence on the model parameters to only a few specific data partitions. This process helps to pinpoint the effects of unlearning to a few data features. But, again, the approach is only effective with a small number of features compared to the size of the dataset. Notably, data privacy and data deletion are not completely correlative [57]. Data privacy does not have to ensure data deletion (e.g., differential privacy), and data deletion does not have to ensure data privacy.

Knowledge Adaptation (Knowledge Distillation). Knowledge adaptation selectively removes to-be-forgotten data samples [28]. In this approach [28], one trains two neural networks as teachers (competent and incompetent) and one neural network as a student. The competent teacher is trained on the complete dataset, while

the incompetent teacher is randomly initialised. The student is initialised with the competent teacher's model parameters. The student is trained to mimic both competent teacher and incompetent teacher by a loss function with KL-divergence evaluation values between the student and each of the two teachers. Notably, the competent teacher processes the retained data and the incompetent teacher deals with the forgotten data.

Beyond Chundwat et al. [28], machine learning models have been quickly and accurately adapted by reconstructing the past gradients of knowledge-adaptation priors in [88]. Ideas similar to knowledge-adaptation priors were also investigated in [57, 202]. Lin et al. [100] introduce ERM-KTP, a knowledge-level machine unlearning framework that leverages knowledge transfer to erase specific learned information while maintaining model performance. Wang et al. [192] propose KGA, a general machine unlearning framework that aligns knowledge gaps between original and unlearned models for efficient data removal. Zhang et al. [215] present a stochastic teacher network method for unlearning, which trains the model to forget specific data points through stochastic processes. Kim et al. [89] develop Layer Attack Unlearning, a fast and accurate method that uses layer-level attack and knowledge distillation to remove sensitive information from machine learning models. Lastly, Bonato et al. [9] investigate the impact of out-of-distribution images on unlearning models, proposing that retaining certain sets of data can help restore performance in unlearned models. In general, knowledge adaptation is applicable to a wide range of unlearning requests and scenarios. However, it is difficult to provide a theoretical guarantee for this approach.

MCMC Unlearning (Parameter Sampling). Sampling-based machine unlearning has also been suggested as a way to train a standard machine learning model to forget data samples from the training data [128]. The idea is to sample the distribution of model parameters using Markov chain Monte Carlo (MCMC). It is assumed that the forgetting set is often significantly smaller than the training data (otherwise retraining might be a better solution). Thus, the parameter distribution $Pr(w_r)$ of the retrained models should not differ much from that of the original model $Pr(w)$. In other words, the posterior density $Pr(w_r|D)$ should be sufficient large for sampling [128]. More precisely, the posterior distribution from the retrained parameters can be defined as:

$$Pr(w_r|D) \approx Pr(w|D) \propto Pr(D|w)Pr(w) \tag{15}$$

Here, the prior distribution $Pr(w)$ is often available from the learning algorithm, which means the stochasticity of learning via sampling can be estimated. The likelihood $Pr(D|w)$ is the prediction output of the model itself, which is also available after training. From Eq. 15, we only know that the density function of $Pr(w|D)$ is proportional to a function $f(w) = Pr(D|w)Pr(w)$, which means $Pr(w|D)$ cannot be directly sampled. This is where MCMC comes into play, as it can still generate the next samples using a proposal density $g(w'|w)$ [128]. However, $g(w'|w)$ is assumed to be a Gaussian distribution centered on the current sample (the sampling process can be initialized with the original model).

As a result, a candidate set of model parameters $Pr(w_r|D)$ is constructed from the sampling, and the unlearning output is calculated by simply maximizing the posterior probability $Pr(w|D_r)$, i.e.:

$$w_r = \arg\max_w Pr(w|D_r) \tag{16}$$

The benefit of such sampling-based unlearning is that no access to the forgetting set is required.

5.2 Model-Intrinsic Approaches

The model-intrinsic approaches include unlearning methods designed for a specific type of models. Although they are model-intrinsic, their applications are not necessarily narrow, as many machine learning models can share the same type.

Unlearning for softmax classifiers (logit-based classifiers). Softmax (or logit-based) classifiers are classification models $M : \mathcal{Z} \rightarrow \mathbb{R}^K$ that output a vector of logits $l \in \mathbb{R}^k$, where K is the number of classes, for each data sample $x \in \mathcal{Z}$. The core task of $M(x)$ is to estimate the probability distribution $Pr(X, Y)$, where X is the random variable in $\mathcal{X}$, and Y is the random variable in $1, \ldots, K$, such that:

$$Pr(Y = i|X = x) \approx \sigma(l_i) \tag{17}$$

Here, $\sigma(l_i) = \frac{\exp(l_i)}{\sum_{j=1..K} \exp l_j}$ is the softmax function. This formulation applies to logistic regression and deep neural networks with a densely connected output layer using softmax activations [4]. Baumhauer et al. [4] proposed an unlearning method for softmax classifiers based on a linear filtration operator to proportionally shift the classification of the to-be-forgetten class samples to other classes. However, this approach is only works for class removal.

Unlearning for linear models. Izzo et al. [79] proposed an approximate unlearning method for linear and logistic models based on influence functions. They approximated a Hessian matrix computation with a project residual update [14, 79] that combines gradient methods with synthetic data. It is suitable for forgetting small groups of points out of a learned model. Some other studies consider an online setting for machine unlearning (aka online data deletion) [57, 99], in which the removal request is a sequence of entries that indicates which data item is to be unlearned. In general, this setting is more challenging than normal setting because indistinguishability must hold for any entry and for the end of the deletion sequence. The goal is to achieve a lower bound on amortized computation time [57, 99].

Li et al. [99] formulated a special case of the online setting where data is only accessible for a limited time so there is no full training process in the first place. More precisely, the system is allowed a constant memory to store historical data or a data sketch, and it has to make predictions within a bounded period of time. Although the data to be forgotten can be unlearned from a model on-the-fly using a regret scheme on the memory, this particular unlearning process is only applicable to ordinary linear regression [99].

Unlearning for Tree-based Models. Tree-based models are classification techniques that partition the feature space recursively, where the features and cut-off thresholds to split the data are determined by some criterion, such as information gain [153, 203]. There is a class of tree-based models, called extremely randomized trees [56], that are built by an ensemble of decision trees. These are very efficient because the candidate set of split features and cut-off thresholds are randomly generated. The best candidate is

selected by a reduction in Gini impurity, which avoids the heavy computation of logarithms.

Schelter et al. [153] proposed an unlearning solution for extremely randomized trees by measuring the robustness of the split decisions. A split decision is robust if removing k data items does not reverse that split. Note that k can be bounded, and it is often small as only one in ten-thousand users who wants to remove their data at a time [153]). The learning algorithm is redesigned such that most of splits, especially the high-level ones, are robust. For the non-robust splits, all subtree variants are grown from all split candidates and maintained until a removal request would revise that split. When that happens, the split is switched to its variant with higher Gini gain. As a result, the unlearning process involves recalculating the Gini gains and updating the splits if necessary.

One limitation of this approach is that if the set to be forgotten is too large, there might be many non-robust splits. This would lead to high storage costs for the subtree variants. However, it does give a parameterized choice between unlearning and retraining. If there are many removal requests, retraining might be the best asymptotically. Alternatively, one might limit the maximum number of removal requests to be processed at a time. Moreover, tree-based models have a highly competitive performance for many predictive applications [153].

Unlearning for Bayesian Models. Bayesian models are probabilistic models that approximate a posterior likelihood [49, 50, 82, 127]. Also known as Bayesian inference, this process is particularly useful when a loss function is not well-defined or does not even exist. Bayesian models cover a wide range of machine learning algorithms, such as Bayesian neural networks, probabilistic graphical models, generative models, topic modeling, and probabilistic matrix factorization [137, 147, 213].

Unlearning for Bayesian models requires a special treatment, as the training already involves optimizing the posterior distribution of the model's parameters. It also often involves optimizing the Kullback-Leibler divergence between a prior belief and the posterior distribution [127]. Nguyen et al. [127] proposed the notion of *exact Bayesian learning*:

$$Pr(w|D_r) = Pr(w|D)Pr(D_f|D_r)/Pr(D_f|w) \propto Pr(w|D)/Pr(D_f|w) \tag{18}$$

where $Pr(w|D_r)$ is the distribution of a retrained model (as if it were trained only on D_r). However, the posterior distribution $Pr(w|D_r)$ can only be sampled directly when the model parameters are discrete-valued (quantized) or the prior is conjugate [127]. For non-conjugate priors, Nguyen et al. [127] proved that we can approximate $Pr(w|D_r)$ by minimizing the KL divergence between $Pr(w|D)$ and $Pr(w|D_r)$. Since $Pr(w|D)$ is the original model's parameter distribution, this approximation prevents catastrophic unlearning. As such, the retained model performs significantly better than the unlearned model in terms of accuracy.

A notion of certified Bayesian unlearning has also been studied, where the KL divergence between the unlearned model and the retrained model is bounded [49, 50, 82]:

$$KL(Pr(A(D_r)), \mathop{\mathbb{E}}_{A(D)} Pr(U(D, D_f, A(D)))) \leq \epsilon \tag{19}$$

Here, the result of the unlearning process is an expectation over the parameter distribution of the original model $A(D) \sim Pr(w|D)$. This certification can be achieved for some energy functions when formulating the evidence lower bound (ELBO) in Bayesian models [49, 50, 82].

Unlearning for DNN-based Models. Deep neural networks are advanced models that automatically learn features from data. As a result, it is very difficult to pinpoint the exact model update for each data item [60–62, 72, 121]. Fortunately, deep neural networks consist of multiple layers. For layers with convex activation functions, existing unlearning methods such as certified removal mechanisms can be applied [14, 64, 126, 154]. For non-convex layers, Golatkar et al. [59, 61] proposed a caching approach that trains the model on data that are known a priori to be permanent. Then the model is fine-tuned on user data using some convex optimization.

Sophisticated unlearning methods for DNNs rely primarily on influence functions [91, 214]. Here, Taylor expansions are used to approximate the impact of a data item on the parameters of black-box models [212]. Some variants include DeltaGrad [201], which stores the historical updates for each data item, and Fisher-based unlearning [60], which we discussed under §5.1). However, influence functions in deep neural networks are not stable with a large forget set [3, 113, 114].

More precisely, after the data to be forgotten has been deleted from database, Fisher-based unlearning [60] works on the remaining training data with the Newton's method, which uses a second-order gradient. To mitigate potential information leaks, noise is injected into the model's parameters [30]. As the Fisher-based method aims to approximate the model without the deleted data, there can be no guarantee that all the influence of the deleted data has been removed. Although injecting noise can help mitigate information leaks, the model's performance may be affected by the noise [30].

Golatkar et al. [60] point out that the Hessian computation in certified removal mechanisms is too expensive for complex models like deep neural networks. Hence, they resorted to an approximation of Hessian via Levenberg-Marquardt semi-positive-definite approximation, which turns out to correspond with the Fisher Information Matrix [117]. Although it does not provide a concrete theoretical guarantee, Fisher-based unlearning could lead to further information-theoretic approaches to machine unlearning [61, 66].

5.3 Data-Driven Approaches

Data Partitioning (Efficient Retraining). The approaches falling into this category uses data partitioning mechanisms to speed up the retraining process. Alternatively, they partially retrain the model with some bounds on accuracy. Bourtoule et al. [10] proposed the well-known SISA framework (Fig. 2) that partitions the data into shards and slices. Each shard has a single model, and the final output is an aggregation of multiple models over these shards. For each slice of a shard, a model checkpoint is stored during training so that a new model can be retrained from an intermediate state [2, 10]. Dukler et al. [41] partition the training data into disjoint shards and builds a graph structure to track dependencies between shards and model updates. When data is requested to be forgotten, the system efficiently removes the effects of specific shards by rolling back and recalculating model updates only for affected shards.

Data Augmentation (Error-manipulation noise). Data augmentation is the process of enriching or adding more data to support a

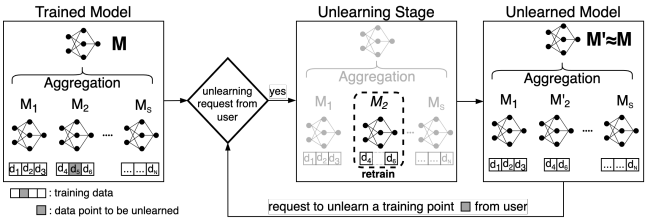

Figure 2: Efficient retraining for machine unlearning using data partition

model's training [208]. Such mechanisms can be used to support machine unlearning as well. Huang et al. [75] proposed the idea of error-minimizing noise, which tricks a model into thinking that there is nothing to be learned from a given set of data (i.e., the loss does not change). However, it can only be used to protect a particular data item before the model is trained. A similar setting was also studied by Fawkes [157], in which a targeted adversarial attack is used to ensure the model does not learn anything from a targeted data item.

Conversely, Tarun et al. [172] proposed error-maximizing noise to impair the model on a target class of data (to be forgotten). However, this tactic does not work on specific data items as it is easier to interfere with a model's prediction on a whole class as opposed to a specific data item of that class [172].

Data influence. This group of unlearning approaches studies how a change in training data impacts a model's parameters [14, 30, 199], where impact is computed using influence functions [29, 114]. However, influence functions depend on the current state of a learning algorithm [199]. To mitigate this issue, several works store a training history of intermediate quantities (e.g., model parameters or gradients) generated by each step of model training [63, 126, 201, 202]. Then, the unlearning process becomes one of subtracting these historical updates. However, the model's accuracy might degrade significantly due to catastrophic unlearning [127] since the order in which the training data is fed matters to the learning model. Moreover, the influence itself does not verify whether the data to be forgotten is still included in the unlearned model [173, 174].

Zeng et al. [212] suggested a new method of modeling data influence by adding regularization terms into the learning algorithm. Although this method is model-agnostic, it requires intervening in the original training process of the original model. Moreover, it is only applicable to convex learning problems and deep neural networks. Peste et al. [138] closed this gap by introducing a new Fisher-based unlearning method, which can approximate the Hessian matrix. This method works for both shallow and deep models, and also convex and non-convex problems. The idea is to efficiently compute the matrix inversion of a Fisher Information Matrix using rank-one updates. However, as the whole process is approximate, there is no concrete guarantee on the unlearned model.

Yamasita et al. [206] propose a one-shot unlearning technique that removes the need for extra training by adding noise to the model's sensitive parameters, estimated using the Fisher information matrix (FIM). Unlike existing methods, this approach does not require retaining training data for FIM calculation. Instead, it utilizes class-specific synthetic signals, or mnemonic codes, during training. Each class is assigned a mnemonic code, which is incorporated into the training data as:

$$\tilde{x_i} = (1 - \lambda)x_i + \lambda\xi_c \tag{20}$$

where $\lambda\xi_c$ represents the mnemonic codes and λ is the hyperparameter. This eliminates the need to retain training data for the unlearning process, reducing storage costs and enhancing practicality.

6 PUBLISHED RESOURCES ON MACHINE UNLEARNING

6.1 Published Unlearning Algorithms

Some implementations of algorithms and models are available that have contributed to baseline experiments in machine unlearning. A summary of published implementations, including their language and platform details, the corresponding models, and URLs to their code repositories, are presented in Table 4.

6.2 Published Datasets

The most widely used datasets in machine unlearning are shown in Table 5. We have classified these into several groups based on their field of application, including image, tabular, text, sequence, and graph. Due to the space limit, the details of these datasets can be found in our technical report [130].

6.3 Evaluation Metrics

The most often used metrics for measuring anomaly detection performance include accuracy, completeness, unlearn time, distance, and forgetting scores. Their formulas and common usage are summarized in Table 6. More detailed descriptions are given below.

Accuracy. In machine unlearning, a model's accuracy needs to be compared on three different datasets: (1) The set to be forgotten. Since the expected behaviour of an unlearned model should mirror that of a retrained model, the accuracy on the remaining data should be similar to the retrained model. (2) The retained set. The retained set's accuracy should be close to that of the original model. (3) The test set. The unlearned model should still perform well on a separate test dataset compared to the retrained model.

Completeness. The influence of the to-be-removed samples on the unlearned model must be completely eliminated. Completeness, hence, measures the degree to which an unlearned model is compatible with a retrained model [12]. If the unlearned model gives similar predictions to a retrained model for all samples, the operation of feeding samples or observing the model's information is impractical to achieve the forgotten data and its lineage. The final metric is often calculated as the overlap of output space (e.g., the Jaccard distance) between the unlearned model and the retraining. However, computing this metric is often exhaustive.

Unlearning time and Retraining time. Timeliness quantifies the time saved when using unlearning instead of retraining for model update. The quicker the system restores privacy, security, and usefulness, the more timely the unlearning process. In particular, retraining uses the whole training set to execute the learning algorithm, whereas unlearning executes the learning algorithm on

Table 4: Published Algorithms and Models

Unlearning Algorithms	Language	Platform	Applicable ML Models	Code Repository
SISA [10]	Python	-	Model-agnostic	https://github.com/cleverhans-lab/machine-unlearning
Athena [166, 167]	Python	-	Model-agnostic	https://github.com/inspire-group/unlearning-verification
AmnesiacML [63]	Python	-	Model-agnostic	https://github.com/lmgraves/AmnesiacML
Kpriors [88]	Python	Pytorch	Model-agnostic	https://github.com/team-approx-bayes/kpriors
ERM [126]	Python	-	Model-agnostic	https://github.com/ChrisWaites/descent-to-delete
ShallowAttack [23]	Python	Pytorch	Model-agnostic	https://github.com/MinChen00/UnlearningLeaks
UnrollingSGD [173]	Python	-	Model-agnostic	https://github.com/cleverhans-lab/unrolling-sgd
DeltaGrad [201]	Python	-	Model-agnostic	https://github.com/thuwuyinjun/DeltaGrad
Amnesia [152]	Rust	-	Model-agnostic	https://github.com/schelterlabs/projects-amnesia
SCAR [9]	Python	-	Model-agnostic	https://github.com/jbonato1/SCAR
KGA [192]	Python	Pytorch	Model-agnostic	https://github.com/Lingzhi-WANG/KGAUnlearn
MUPy [12]	Python	LensKit	kNN	https://github.com/theLauA/MachineUnlearningPy
DelKMeans [57]	Python	-	kMeans	https://github.com/tginart/deletion-efficient-kmeans
CertifiedRem [64]	Python	Pytorch	Linear models	https://github.com/facebookresearch/certified-removal
CertAttack [116]	Python	Tensorflow	Linear models	https://github.com/ngmarchant/attack-unlearning
PRU [79]	Python	-	Linear models	https://github.com/zleizzo/datadeletion
DeltaBoost [203]	Python	-	Tree-based models	https://github.com/Xtra-Computing/DeltaBoost
HedgeCut [153]	Python	-	Tree-based models	https://github.com/schelterlabs/hedgecut
DaRE-RF [11]	Python	-	Tree-based models	https://github.com/jjbrophy47/dare_rf
MCMC-Unlearning [49]	Python	Pytorch	Bayesian models	https://github.com/fshp971/mcmc-unlearning
BIF [50]	Python	Pytorch	Bayesian models	https://github.com/fshp971/BIF
L-CODEC [121]	Python, Matlab	Pytorch	Deep learning	https://github.com/vsingh-group/LCODEC-deep-unlearning
SelectiveForgetting [60, 61]	Python	-	Deep learning	https://github.com/AdityaGolatkar/SelectiveForgetting
Neurons [33]	Python	-	Deep learning	https://github.com/Hunter-DDM/knowledge-neurons
Unlearnable [75]	Python	-	Deep learning	https://github.com/HanxunH/Unlearnable-Examples
DLMA [208]	Python	-	Deep learning	https://github.com/AnonymousDLMA/MI_with_DA
ERM-KTP [100]	Python	Pytorch	Deep learning	https://github.com/RUIYUN-ML/ERM-KTP
GraphProjector [32]	Python	-	Graph Learning	https://github.com/CongWeilin/Projector
GraphEditor [30]	Python	-	Graph Learning	https://anonymous.4open.science/r/GraphEditor-NeurIPS22-856E/README.md
FedLU [219]	Python	Pytorch	Graph Learning	https://github.com/nju-websoft/FedLU/
GUIDE [189]	Python	-	Graph Learning	https://github.com/Happy2Git/GUIDE
GNNDelete [26]	Python	-	Graph Learning	https://github.com/mims-harvard/GNNDelete
GraphEraser [24]	Python	-	Graph Learning	https://github.com/MinChen00/Graph-Unlearning
GST-Unlearn [132]	Python	-	Graph Learning	https://zenodo.org/records/7613150
RecEraser [18]	Python, C++	-	Recommender Systems	https://github.com/chenchongthu/Recommendation-Unlearning
ADV-MULTVAE [51]	Python	-	Recommender Systems	https://github.com/CPJKU/adv-multvae
FedEraser [103]	Python	-	Federated Learning	https://www.dropbox.com/s/1lhx962axovbbom/FedEraser-Code.zip?dl=0
RapidFed [109]	Python	-	Federated Learning	https://github.com/yiliucs/federated-unlearning
SIFU [47]	Python	PyTorch	Federated Learning	https://github.com/Accenture/Labs-Federated-Learning/tree/SIFU
Fast-FedUL [78]	Python	-	Federated Learning	https://github.com/thanhtrunghuynh93/fastFedUL
FATS [171]	Python	-	Federated Learning	https://github.com/Happy2Git/FATS_supplement
EUL [20]	Python	-	LLM	https://github.com/SALT-NLP/Efficient_Unlearning
E2URec [190]	Python	Pytorch	LLM, Recommender Systems	https://github.com/justarter/E2URec
Ext-Sub [73]	Python	-	LLM	https://github.com/HITsz-TMG/Ext-Sub
SP [139]	Python	-	LLM	https://github.com/nickypro/selective-pruning
Quark [112]	Python	-	LLM	https://github.com/GXimingLu/Quark
I2I-Unlearn [96]	Python	Pytorch	Generative Models	https://github.com/jpmorganchase/i2i-generator-unlearning
AdvUnlearn [216]	Python	-	Generative Models	https://github.com/OPTML-Group/AdvUnlearn
FAST [133]	Python	-	Generative Models	https://github.com/Subhodip123/weak-unlearning-gan

-: No Dedicated Platforms.

a limited amount of summations; hence, the speed of unlearning is quicker due to the reduced size of the training data.

Relearn time. Relearning time is an excellent proxy for measuring the amount of unlearned data information left in the model. If a model recovers its performance on unlearned data with just a few steps of retraining, it is extremely probable that the model has retained some knowledge of the unlearned data.

The layer-wise distance. The weight difference between the original and unlearned neural networks helps when evaluating the unlearning impact on each layer [172]. The weight difference should be comparable to a retrained model given that a shorter distance indicates ineffective unlearning. Likewise, a much longer distance may point to a Streisand effect and possible information leaks.

Activation Distance. The activation distance is the separation between the final activation of the scrubbed weights and the retrained model. A shorter activation distance indicates superior unlearning.

JS-Divergence. When paired with the activation distance, the JS-Divergence between the predictions of the unlearned and retrained model provides a more full picture of unlearning. Less divergence results in better unlearning. The formula of JS-Divergence is

$$\mathcal{JS}(M(x), T_d(x)) = 0.5 * \mathcal{KL}(M(x)||m) + 0.5 * \mathcal{KL}(T_d(x)||m)$$

where M is unlearned model, T_d is a competent teacher, and $\mathcal{KL}$ is The Kullback-Leibler divergence [92], $m = \frac{M(x)+T_d(x)}{2}$.

Membership Inference. The membership inference metric leverages a membership inference attack to determine whether or not any information about the forgotten samples remains in the model [23]. The set to be forgotten should have reduced the attack probability

Table 5: Published Datasets (URLs are embedded in dataset names).

Category	Dataset	#Items	Disk Size	Downstream Applications	REF
Image	MNIST	70K	11MB	Classification	29+ papers ([10, 57, 64], ...)
	CIFAR	60K	163MB	Classification	16+ papers ([63, 75, 159, 191], ...)
	SVHN	600K	400MB+	Classification	8+ papers([10, 64, 75, 76, 105], ...)
	LSUN [209]	69M+	1TB+	Classification	[64]
	ImageNet [35]	14M+	166GB	Classification	[10, 53, 72, 75, 166, 167]
Tabular	Adult	48K+	10MB	Classification	8+ papers([11, 23, 103, 108, 153, 161], ...)
	Breast Cancer	569	$< 1MB$	Classification	[52, 199]
	Diabetes	442	$< 1MB$	Regression	[11, 25, 196]
Text	IMDB Review	50K	66MB	Sentiment Analysis	[107]
	Reuters	11K+	73MB	Categorization	[107]
	Newsgroup	20K	1GB+	Categorization	[107]
Sequence	Epileptic Seizure	11K+	7MB	Timeseries Classification	[28]
	Activity Recognition	10K+	26MB	Timeseries Classification	[28]
	Botnet	72M	3GB+	Clustering	[57]
Graph	OGB	100M+	59MB	Classification	[27, 31]
	Cora	2K+	4.5MB	Classification	[22, 27, 31]
	MovieLens	1B+	3GB+	Recommender Systems	[152]

REF: Papers that run experiments on the dataset.

in the unlearned model. The chance of an inference attack should be reduced in the unlearned model compared to the original model for the forgotten class data.

ZRF (Zero Retrain Forgetting) score. ZRF makes it possible to evaluate unlearning approaches independent of retraining [28]. The unpredictability of the model's predictions is measured by comparing them to an unskilled instructor. ZRF compares the set to be forgotten's output distribution to the output of a randomly initialised model, which in most situations is our lousy instructor. The ZRF score ranges between 0 and 1; it will be near to 1 if the model's behaviour with the forgotten samples is entirely random, and close to 0 if it exhibits a certain pattern. Formally, $\mathcal{ZRF} = 1 - \frac{1}{nf}\sum_{i=0}^{n_f} \mathcal{JS}(M(x_i), T_d(x_i))$, where x_i is the i_{th} sample from the set to be forgotten with a total number of samples n_f.

Anamnesis Index (AIN). AIN values range between 0 and 1. The better the unlearning, the closer to 1. Instances where information from the classes to be forgotten are still preserved in the model correlate to AIN levels well below 1. A score closer to 0 also suggests that the unlearned model will rapidly relearn to generate correct predictions. This may be due to the fact that the last layers contain limited reversible modifications, which degrades the performance of the model on the forgotten classes. If an AIN score is much greater than 1, it may suggest that the approach causes parameter changes that are so severe that the unlearning itself may be detected (Streisand effect). This might be due to the fact that the model was pushed away from the original point and, as a result, is unable to retrieve previously learned knowledge about the forgotten class(es). The formula for calculating an AIN value is $AIN = \frac{r_t(M_u, M_{orig}, \alpha)}{r_t(M_s, M_{orig}, \alpha)}$, where $\alpha\%$ is a margin around the initial precision used to determine relearn time. $r_t(M, M_{orig}, \alpha)$ are mini-batches (or steps) to be achieved by the model M on the classes to be forgotten within $\alpha\%$ of the precision compared to the original model M_{orig}. M_u and M_s respectively represent the unlearned model and a model trained from scratch.

Epistemic Uncertainty. Epistemic uncertainty is an uncertainty metric that measures how much we know about the optimal hypothesis in the parameter space [77]. That is, are we certain that the current model parameters are optimal for a given dataset?. The influence functions are computed as the trace of Fischer Information Matrix (FIM) [5]:

$$i(w; D) = tr(\mathcal{I}(w; D)) \tag{21}$$

where $\mathcal{I}(w; D)$ is the FIM that determines the quantities of information that the model parameters w hold regarding the dataset D [5], which can be approximate via Levenberg-Marquart [117]:

$$\mathcal{I}(w; D) \approx \frac{1}{|D|} \sum_{x,y \in D} \left(\frac{\partial \log p(y|x; w}{\partial w} \right)^2 \tag{22}$$

Becker et al. [5] proposed an efficacy score based on the epistemic uncertainty as follows.

$$\text{efficacy}(w; D) = \begin{cases} \frac{1}{i(w;D)}, \text{if i(w;D)} > 0 \\ \infty, \text{otherwise} \end{cases} \tag{23}$$

It measures how much information the model exposes. This metric is computationally efficient and does not require access to the retrained model [5]. A better unlearning algorithm will produce the unlearned model with lower efficacy score. However, it only measures the overall information reduction, not the specific information related to the data to be forgotten. Moreover, it should be used along with an accuracy metric, as less exposing models do not necessarily have good predictive performance [5].

Model Inversion attack. It has been shown that model inversion attacks may reproduce the records from trained machine learning models. It is possible to replicate samples of target points around a regression value if white-box access to a trained model is available. The data provided from the unlearned model following inversion should not include information about the forget class. This metric is qualitative and often used in image applications [63].

7 UNLEARNING APPLICATIONS

7.1 Unlearning in Recommender Systems

In the field of machine learning, recommender systems are used to predict what a user might want to buy or watch. They often use collaborative filtering to learn a user's preferences based on their past behavior. However, a recommender system may be required to

Table 6: Evaluation Metrics

Evaluation Metrics	Formula/Description	Usage	REF
Accuracy	Accuracy on unlearned model on forget set and retrain set	Evaluating the predictive performance of unlearned model	[29, 59, 60, 172]
Completeness	The overlapping (e.g. Jaccard distance) of output space between the retrained and the unlearned model	Evaluating the indistinguishability between model outputs	[12]
Unlearn Time	The amount of time of unlearning request	Evaluating the unlearning efficiency	[12]
Relearn Time	The epochs number required for the unlearned model to reach the accuracy of source model	Evaluating the unlearning efficiency (relearn with some data sample)	[12, 172]
Layer-wise Distance	The weight difference between original model and retrain model	Evaluate the indistinguishability between model parameters	[172]
Activation Distance	An average of the L2-distance between the unlearned model and retrained model's predicted probabilities on the forget set	Evaluating the indistinguishability between model outputs	[28, 59]
JS-Divergence	Jensen-Shannon divergence between the predictions of the unlearned and retrained model: $\mathcal{JS}(M(x), T_d(x)) = 0.5 * \mathcal{KL}(M(x)\|\|m) + 0.5 * \mathcal{KL}(T_d(x)\|\|m)$	Evaluating the indistinguishability between model outputs	[28]
Membership Inference Attack	Recall (#detected items / #forget items)	Verify the influence of forget data on the unlearned model	[59, 63]
ZRF score	$\mathcal{ZRF} = 1 - \frac{1}{nf}\sum_{i=0}^{n_f}\mathcal{JS}(M(x_i), T_d(x_i))$	The unlearned model should not intentionally give wrong output ($\mathcal{ZRF} = 0$) or random output ($\mathcal{ZRF} = 1$) on the forget item	[28]
Anamnesis Index (AIN)	$AIN = \frac{r_t(M_u, M_{orig}, \alpha)}{r_t(M_s, M_{orig}, \alpha)}$	Zero-shot machine unlearning	[29]
Epistemic Uncertainty	$\text{efficacy}(w; D) = \begin{cases} \frac{1}{i(w;D)}, \text{if i(w;D)} > 0 \\ \infty, \text{otherwise} \end{cases}$	How much information the model exposes	[5]
Model Inversion Attack	Visualization	Qualitative verifications and evaluations	[63]

REF: Highlighted papers that used/proposed the metric

forget private training points and its complete impact on a model in order to protect user privacy or comply with explicit user removal requests. Utility is another reason for unlearning requests. For example, the accuracy of a recommendation can be degraded due to out-of-distribution data or poisoning attacks [38, 116]. In the latter case, data that is detected as poisoned will need to be removed, while in the former case, old data may need to be removed so that the system keeps up with the new data distribution.

Unlearning techniques for machine learning in general cannot be used directly on recommender systems [18, 188]. For example, collaborative filtering recommendation uses the information of similarity across users-item interaction; and thus, arbitrarily partitioning the training sets could break such coupling information [10]. Some researchers have developed unlearning methods for graph data [22] and while recommendation data can be modelled as graphs, their user-item interactions are not uniform [18].

To overcome these challenges, Chen et al. [18] proposed a partition-based retraining process, called smart retraining, to unlearn the model from the removed user behavior data. The idea is to develop a strategy to partition the data with regard to the resemblance between users and items while maintaining the balance between different partitions for retraining. Next, the output of the submodels is combined using an attention-based method, each of which is associated with each disjoint partition.

In other settings, Yuan et al. [210] propose FRU, a method to erase user contributions from federated learning systems using a rollback mechanism. AltEraser [104] is a fast unlearning technique for neural recommender systems that employs second-order optimization to efficiently remove unwanted data without full retraining. Sinha et al. [165] examine unlearning in multi-modal recommendation systems, addressing the complexities of handling different data types and reducing legal risks while optimizing computational efficiency. Liu et al. [98] explore attribute-wise unlearning to make users indistinguishable in recommender systems, enhancing privacy by eliminating sensitive user attributes.

7.2 Unlearning Federated Learning

Recently, federated learning has become popular in the field of machine learning [119]. One typical federated learning scenario is building a machine learning model from healthcare information. Due to privacy regulations, the medical record data cannot leave the clients' devices. Here, the clients could be hospitals or personal computers and are assumed to have machine learning environments. The server does not transmit the actual data to the global model. Rather, there is a communication protocol between the clients and servers that governs collaborative model training [197]. In the literature, the communication protocol Federated Average (FedAvg) [119] is typically used for model training. It consists of multiple rounds. Each round, the current global model weights are transmitted to the clients. Based on these weights, each client uses stochastic gradient descent to adjust their local model. Then the local model's weights are forwarded to the server. In the final phase of each loop, the server aggregates the received weights by (weighted) averaging to prepare the global model for the next round [68].

Given such training protocols, machine unlearning cannot be extended easily to the federated learning setting [47, 78, 171]. This

is because the global weights are computed by aggregations rather than raw gradients. These are especially mixed up when many clients participate [53]. Moreover, these clients might have some overlapping data, making it difficult to quantify the impact of each training item on the model weights [107]. Using classic unlearning methods by gradient manipulation may even lead to severe accuracy degradation or new privacy threats [103].

Additionally, current studies on federated unlearning tend to assume that the data to be removed belongs wholly to one client [103, 107, 191, 197]. With this assumption, the historical contributions of particular clients to the global model's training can be logged and erased easily. However, erasing historical parameter updates might still damage the global model, but there are many strategies for overcoming this issue. For example, Liu et al. [103] proposed calibration training to separate the individual contributions of clients as much as possible. This mechanism does not work well for deep neural networks, but it does work well with shallow architectures such as a 2-layer CNN or a network with two fully-connected layers. In addition, there is a trade-off between scalability and precision due to the cost of storing historical information on the federated server. Wu et al. [197] put forward a knowledge distillation strategy that uses a prime global model to train the unlearned model on the remaining data. However, as the clients' data is not accessible by the server, some unlabeled (synthetic) data that follows the distribution of the whole dataset needs to be sampled and extra rounds of information exchange are needed between the clients and server. As a result, the whole process is costly and approximate. Also, it might be further offset when the data is non-IID [109]. On another spectrum, Liu et al. [109] proposed a smart retraining method for federated unlearning without communication protocols. The approach uses the L-BFGS algorithm [6, 8] to efficient solve a Hessian approximation with historical parameter updates for global model retraining. However, this method is only applicable to small models ($\leq$ 10K parameters). Plus, it involves storing old model snapshots (including historical gradients and parameters), which poses some privacy threats.

Recently, Shaik et al. [155] propose a framework, namely FRAMU, that uses reinforcement learning to help decentralized agents make optimal data unlearning decisions based on real-time feedback. By leveraging attention mechanisms and balancing exploration and exploitation, the framework ensures continual model adaptation, improving convergence, privacy preservation, and decision-making in dynamic, multi-modal environments [93]. Further studies on federated unlearning can be found at [110, 140, 158, 200].

7.3 Unlearning in Graph Embedding

So far the data in machine unlearning settings is assumed to be independent. However, there are many cases where the data samples are relational, such as is the case with graph data. Graph representation learning is a well-established research direction in machine learning, specifically in deep learning [19, 69, 94, 111]. However, applying machine unlearning to graph representation learning is arguably more challenging. First, the data is correlated, and it is non-trivial to partition the data, even uniformly. Second, the unlearning requests can happen upon a node or an edge. Third, the graph data itself might be non-uniform due to unbalanced connected components in the graph. Therefore, existing graph partition methods might lead to unbalanced data partitions, making the retraining process non-uniform.

To mitigate these problems, Chen et al. [22] proposed a new graph partitioning strategies especially for machine unlearning. The general idea is based on the notion of assignment preference that represents the benefit of a node assigned to a shard (i.e., a data partition). Such node-shard pairs are further fine-tuned with neighbor counts, which track down the number of neighbors of a node belonging to the same target shard. The authors also proposed an aggregation method to combine different partition strategies. Further, the retraining process is based on message passing in graph neural networks, which facilitates fast retraining.

Unlearning without retraining is also possible for graph embedding. However, several challenges need to be overcome. The interdependency of graph data, especially across different subgraphs, is non-trivial for model training. A removal of node and edge could not only cause an impact on its neighbor but also on multi-hops. Cong et al. [30, 31] proposed a one-shot unlearning solution that only requires access to the data to be forgotten. The idea is inspired by the architecture of a linear graph neural network (GNN), in which non-linearities in a typical GNN are replaced by a single weight matrix between consecutive convolutional layers. Despite its linear span over all input node features, such linear-GNNs have shown competent performance, e.g. SGC [198] and APPNP [55]. Using this property, Cong et al. [30, 31] proposed an exact unlearning process at the algorithmic level based on linear operations such as projection and recombination.

Existing graph unlearning methods have two main limitations: they either degrade model performance by altering weights shared across all nodes or fail to effectively delete edges due to reliance on local neighborhoods. To solve this, Cheng et al. [26] introduce GNNDELETE, which focuses on two components: Deleted Edge Consistency and Neighborhood Influence. Deleted Edge Consistency ensures that the impact of removed elements is eliminated from both model weights and adjacent representations. Neighborhood Influence retains residual knowledge even after elements are removed, allowing GNNDELETE to modify representations to delete nodes and edges while preserving the remaining knowledge.

Recently, Li et al. [97] propose a mutual evolution framework for general graph unlearning, which iteratively updates both graph structures and model parameters to improve unlearning effectiveness. Zhu et al. [219] introduce a method for heterogeneous federated knowledge graph embedding learning and unlearning, enabling privacy-preserving updates in distributed knowledge graph systems. Wang et al. [189] present inductive graph unlearning, which removes nodes and edges from graph models while maintaining model integrity and scalability in dynamic environments. Finally, Pan et al. [132] investigate unlearning in graph classifiers with limited data, developing techniques that minimize the impact of unlearning on model performance despite constrained resources.

7.4 Unlearning in Lifelong Learning

Unlearning is not always a bad thing for the accuracy of a machine learning model. Machine unlearning has been researched as

a countermeasure against catastrophic forgetting in deep neural networks [38, 101, 136]. Catastrophic forgetting is a phenomenon where deep neural networks perform badly after learning too many tasks [90]. One naive solution to this problem is training the model on the historical data again. Clearly, this solution is impractical not only due to computational cost but also because there is no guarantee that the model will converge and nor is there a guarantee that the forgetting will not happen again [135]. Du et al. [38] suggested a solution based on unlearning to prevent catastrophic forgetting. The core idea is to unlearn harmful samples (e.g., false negatives/positives) and then update the model so that its performance before the forgetting effect is maintained.

Unlearning has also been used to handle exploding losses in machine learning. Here, the term loss involves the computation of $-\log Pr(x)$, and, when $Pr(x)$ is approximately zero, the loss may be arbitrarily significant. The problem is more severe in anomaly detection where normal samples can have very small $Pr(x)$ (abnormal samples have very large $Pr(x)$ and their sum of probabilities is one). Du et al. [38] hence proposed an unlearning method to mitigate this problem with an unlearning loss that regularizes those extreme cases.

Unlearning has been studied for other lifelong settings as well [136]. These setting use incremental models, such as decision tree and naive Bayes, which allows the model to unlearn data samples on-the-fly. Liu et al. [101] considered requests on unlearning specific tasks for lifelong models. In particular, there are three types of requests in lifelong learning: (i) to learn a task permanently, (ii) to learn a task temporarily and forget it later upon a privacy request, and (iii) to forget a task. Different from traditional machine unlearning, unlearning in lifelong learning needs to maintain knowledge transfer between tasks but also preserve all knowledge for the remaining tasks. Moreover, the setting is more challenging as it depends on the order of tasks as the tasks are learnt online during the model lifetime. Additionally, the model cannot keep all previous data (zero-glance unlearning), making the unlearning process more challenging. Liu et al. [101] proposed a solution inspired by SISA, the data partitioning mechanism for smart retraining [10]. It creates an isolated temporary model for each task and merges the isolated models into the main model.

7.5 Unlearning in Large Language Models

Unlearning in large language models (LLMs) has become essential due to increasing privacy concerns and the need to comply with regulations like GDPR [115] and CCPA [134]. As LLMs are trained on vast datasets that may include sensitive information, there is a risk of these models inadvertently memorising and reproducing private data [83, 193, 218]. Chen et al. [20] introduce an efficient framework for unlearning in LLMs that integrates lightweight unlearning layers into transformer models, allowing for the selective removal of specific data without retraining the entire model. They also introduce a fusion mechanism to combine these layers when multiple unlearning requests are made. Kassem [86] propose another unlearning approach called "DeMem", which uses a reinforcement learning feedback loop with a negative similarity score as a reward. This approach incentivises the language model to paraphrase memorised content, thereby reducing the risk of sensitive information being exposed. Wang et al. [190] introduce E2URec, an unlearning method for LLM-based recommender systems. It addresses forgetting user data while preserving model performance by updating a small set of parameters using low-rank adaptation modules and employing a teacher-student framework.

Some works, use machine unlearning as a way to improve the truthfulness and detoxification of LLMs. Hu et al. [73] propose the "Extraction-before-Subtraction" (Ext-Sub) method with parameter-efficient modules (PEMs) to isolate and remove undesirable features like untruthfulness or toxicity, while preserving the model's core capabilities. The method extracts deficiency capabilities from "anti-expert" PEMs and subtracts them from "expert" PEMs, enhancing performance without compromising fundamental abilities. Pochinkov et al. [139] introduce another method by selectively pruning neurons responsible for specific behaviours, such as coding or toxic language, while maintaining overall performance. Lu et al. [112] presents a framework called Quark that uses reinforcement learning as a machine unlearning approach to mitigate undesirable text generation behaviors, such as toxicity and repetition, while preserving language quality.

7.6 Unlearning in Generative Models

Machine unlearning has been studied for generative models, such as diffusion models and adversarial networks. Zhang et al. [216] propose a method that ensures robust concept erasure in diffusion models. This approach uses adversarial training to defend against unwanted model behavior while selectively erasing learned concepts, maintaining the model's overall functionality. Li et al. [96] develop techniques to remove specific data representations from image generators. Unlearncanvas [217], a stylised image dataset to benchmark unlearning performance in diffusion models, has been developed. This dataset provides a standard for evaluating the effectiveness of various unlearning techniques in visual data, emphasizing the need for robust testing frameworks. Panda et al. [133] present FAST, a weak unlearning method for black-box generative models that uses feature similarity to remove specific learned patterns without significantly impacting model accuracy. Tiwary et al. [178] explore a strategy that exploits parameter space semantics in GANs to improve unlearning efficiency, enabling models to adapt to new data while effectively forgetting outdated or undesired information.

8 DISCUSSION AND FUTURE PROSPECTS

In this section, we analyze the current and potential developments in machine unlearning and summarize our findings. In addition, we identify a number of unanswered research topics that could be addressed to progress the foundation of machine unlearning.

8.1 Summary and Trends

Influence functions are dominant methods. Understanding the impact of a given data item on a model's parameters or model performance is the key to machine unlearning [3, 91]. Such insights will speed-up the unlearning process immensely by simply

reversing the model updates associated with the target data. Although there could be some offset in doing so, promising results have shown that this offset can be bounded [29, 113, 114].

Reachability of model parameters. Existing works define unlearning as obtaining a new model with an accuracy as good as if the model had retrained without the data to be forgotten [60, 61, 63, 174]. We argue that such a parameter-space assumption should be taken into serious considerations. As model parameters can be reachable with or without some given data, is there any case where the original and unlearned models share the same parameters? [174] Although some studies use parameter distribution to bound the problem [64], there could still be false positive cases, where some effects from the forgotten data still exist in the unlearned model.

Unlearning verification (Data auditing) is needed. Unlearning verification (or data auditing) is the process of determining whether specific data have been eliminated from a model. To fully enable regulations over the right to be forgotten, the unlearning effect should be independently verified. There have been only a few works on unlearning verification [53, 76]. However, the definition of a successful verification is still controversial as different unlearning solutions use different evaluation metrics, especially when the cutting threshold of a verification metric still depends on the application domain [174].

Federated unlearning is emerging. Federated learning brings about a unique setting for machine unlearning research [103, 107, 191, 197]. It has separate clients participating in the federated training process. As a result, removing a client out of the federation could be done precisely using historical updates. The rational behind this is that the user data on a client mostly helps to make correct predictions about that user. This locality helps to avoid a catastrophic unlearning phenomenon in a traditional machine learning setting. However, we all need to be aware that there are many cases in federated learning where the data is non-IID or the removal request only covers part of the client data.

Model repair via unlearning. Machine learning models can be poisoned by adversarial attacks [106, 187]. Intuitively, if the poisonous data is detected and removed and then the model is retrained, the new model should be poison-free. However, the retraining would be too expensive. This is indeed similar to the unlearning setting. Compared to existing defence methods, the models in machine learning determine then update the inner problematic weights through influence functions.

A similar application is to remove bias from the model due to some biased feature in the data [36, 37]. Status quo studies on fairness and de-biasing learning mostly focus on learning a fair and unbiased feature representation [125, 141, 164, 195], where, machine unlearning, e.g. feature unlearning [66], would ensure the biased features are deleted properly but the model's quality would still be maintained.

In another setting, machine unlearning can be used to repair overtrained deep neural networks by actively unlearning useless, obsolete, or redundant data samples that could cause catastrophic forgetting [38, 60]. Moreover, machine unlearning might be used to boost the model's accuracy as well [2], e.g. as forgetting is similar to compressing in information bottleneck theory [162, 176, 177] [3].

8.2 Open Research Questions

There are several open research questions that future studies can address. This section will list and discuss those fundamental topics in machine unlearning.

Unified Design Requirements. Among the current unlearning approaches, there is no absolute winner that satisfies all design requirements. Most unlearning algorithms focus on approximate unlearning scenarios and data item removal (Table 3). However, there are other types of practical unlearning scenarios that need to be considered, such as zero-glance, zero-shot, few-shot learning. Likewise, there are other types of removal requests that must be handled, e.g., feature removal, class removal, task removal, stream removal, and so on. Moreover, satisfying all design requirements – completeness, timeliness, accuracy, etc. – would make unlearning solutions more applicable to industry-grade systems.

Unified Benchmarking. Although there have been many works on machine unlearning recently, not many of them have a common setting for benchmarking comparisons. In particular, there is not a lot of published source code (Table 4) and each of them targets different learning algorithms or different applications (e.g. recommender systems, graph embedding). Schelter et al. [152] undertook an empirical study but the benchmark was limited to decremental learning methods and focused only on efficiency.

Adversarial Machine Unlearning. More studies have focused on attacking ML systems to improve our understanding and protection of these systems [142, 183, 187]. Adversarial machine unlearning examines attacks on unlearning algorithms to better certify unlearned models [52, 116]. Unlike machine unlearning, which mitigates adversarial attacks [106], adversarial machine unlearning is stricter, addressing not only model accuracy but also privacy guarantees. For instance, it may lack knowledge of learning algorithms but still access the unlearning process.

Interpretable Machine Unlearning. In the future, explanations for machine unlearning can be used to increase confidence in human-AI interactions and enable unlearning verification or removed data auditing [53, 76]. However, the inverted nature of machine unlearning might pose problems for explanation methods to be applicable at all. Devising techniques aimed at explaining the unlearning process (e.g. using influence functions) is still an unsolved task [3, 91].

Machine Unlearning in Evolving Data Streams. Evolving data streams pose problems to machine learning models, especially neural networks, due to shifts in the data distributions and the model predictions [70]. Although there are ways to overcome this limitation [40], they rely on the changes in the model parameters to detect concept drift [70]. However, such detection might be not reliably correct in unlearning settings, where the changes in model parameters are approximate. Consequently, it is expected that machine learning for streaming removal request may attract more attention in the next few years. It is noteworthy that unlearning might be used to repair obsolete models by forgetting old data that

[2] https://insights.daffodilsw.com/blog/machine-unlearning-what-it-is-all-about
[3] https://github.com/ZIYU-DEEP/Awesome-Information-Bottleneck

contradicts with the detected concept drift. However, this requires a contradiction analysis between old and new data [74].

A similar setting is the consideration of out-of-distribution (OOD) data in the forget set. For those settings, the unlearning is imbalanced. Some data might have no impact while the others have great influence on the model parameters. There are studies on learning algorithms for OOD data in federated learning [151]. Hence, it may be worthwhile investigating novel unlearning algorithms tailored to OOD data.

Causality in Machine Unlearning. There are cases where a large amount of data need to be removed from an machine learning system, even though the portion of data to be forgotten is insignificant in comparison to all the data. For example, a data pollution attack might affect millions of data items, but only a few of them can be detected by human experts or SOTA detection methods [13]. Causality analysis [65] could become a useful tool to automatically unlearn the polluted data in this setting and guarantee the non-existence of the polluted information in the final model.

9 CONCLUSION

This survey is the first to investigate machine unlearning techniques in a systematic manner. In this paper, we addressed the primary difficulties and research advancements in conceptualizing, planning, and solving the problems of machine unlearning. In addition, we presented a unified taxonomy that divides machine unlearning strategies into three approaches: model-agnostic methods, model-intrinsic methods, and data-driven methods. We hope that our taxonomy can help categorize future studies and gain deeper insight into methodologies as well as address the difficulties in machine unlearning. Also, we expect this survey can assist researchers in identifying the most optimal unlearning strategies for different applications. The survey provides clear summaries and comparisons between various unlearning methodologies, giving a comprehensive and general view of current work as well as the current process of machine unlearning.

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *SIGSAC*. 308–318.

[2] Nasser Aldaghri, Hessam Mahdavifar, et al. 2021. Coded machine unlearning. *IEEE Access* 9 (2021), 88137–88150.

[3] Samyadeep Basu, Phil Pope, and Soheil Feizi. 2021. Influence Functions in Deep Learning Are Fragile. In *ICLR*.

[4] Thomas Baumhauer, Pascal Schöttle, and Matthias Zeppelzauer. 2020. Machine unlearning: Linear filtration for logit-based classifiers. *arXiv preprint arXiv:2002.02730* (2020).

[5] Alexander Becker and Thomas Liebig. 2022. Evaluating Machine Unlearning via Epistemic Uncertainty. (2022).

[6] Albert S Berahas, Jorge Nocedal, et al. 2016. A multi-batch L-BFGS method for machine learning. *NIPS* 29 (2016).

[7] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. 2012. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *ITCS*. 326–349.

[8] Raghu Bollapragada, Jorge Nocedal, Dheevatsa Mudigere, Hao-Jun Shi, and Ping Tak Peter Tang. 2018. A progressive batching L-BFGS method for machine learning. In *ICML*. 620–629.

[9] Jacopo Bonato, Marco Cotogni, and Luigi Sabetta. 2024. Is Retain Set All You Need in Machine Unlearning? Restoring Performance of Unlearned Models with Out-Of-Distribution Images. *arXiv preprint arXiv:2404.12922* (2024).

[10] Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. 2021. Machine unlearning. In *SP*. 141–159.

[11] Jonathan Brophy and Daniel Lowd. 2021. Machine unlearning for random forests. In *ICML*. 1092–1104.

[12] Yinzhi Cao and Junfeng Yang. 2015. Towards making systems forget with machine unlearning. In *2015 IEEE Symposium on Security and Privacy*. 463–480.

[13] Yinzhi Cao, Alexander Fangxiao Yu, Andrew Aday, Eric Stahl, Jon Merwine, and Junfeng Yang. 2018. Efficient repair of polluted machine learning systems via causal unlearning. In *ASIACCS*. 735–747.

[14] Zihao Cao, Jianzong Wang, Shijing Si, Zhangcheng Huang, and Jing Xiao. 2022. Machine Unlearning Method Based On Projection Residual. In *DSAA*. 1–8.

[15] Gert Cauwenberghs et al. 2000. Incremental and decremental support vector machine learning. *NIPS* 13 (2000).

[16] Yi Chang, Zhao Ren, Thanh Tam Nguyen, Wolfgang Nejdl, and Björn W Schuller. 2022. Example-based Explanations with Adversarial Attacks for Respiratory Sound Analysis. In *INTERSPEECH*.

[17] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. 2011. Differentially private empirical risk minimization. *JMLR* 12, 3 (2011).

[18] Chong Chen, Fei Sun, Min Zhang, and Bolin Ding. 2022. Recommendation unlearning. In *WWW*. 2768–2777.

[19] Fenxiao Chen, Yun-Cheng Wang, Bin Wang, et al. 2020. Graph representation learning: a survey. *ATSIP* 9 (2020).

[20] Jiaao Chen and Diyi Yang. 2023. Unlearn What You Want to Forget: Efficient Unlearning for LLMs. In *EMNLP*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). 12041–12052.

[21] Kongyang Chen, Yao Huang, et al. 2021. Machine unlearning via GAN. *arXiv preprint arXiv:2111.11869* (2021).

[22] Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. 2021. Graph unlearning. *arXiv preprint arXiv:2103.14991* (2021).

[23] Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. 2021. When machine unlearning jeopardizes privacy. In *SIGSAC*. 896–911.

[24] Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. 2022. Graph unlearning. In *CCS*. 499–513.

[25] Yuantao Chen, Jie Xiong, Weihong Xu, and Jingwen Zuo. 2019. A novel online incremental and decremental learning algorithm based on variable support vector machine. *Cluster Computing* 22, 3 (2019), 7435–7445.

[26] Jiali Cheng, George Dasoulas, Huan He, Chirag Agarwal, and Marinka Zitnik. 2023. GNNDelete: A General Strategy for Unlearning in Graph Neural Networks. In *ICLR*.

[27] Eli Chien, Chao Pan, et al. 2022. Certified Graph Unlearning. *arXiv preprint arXiv:2206.09140* (2022).

[28] Vikram S Chundawat, Ayush K Tarun, Murari Mandal, and Mohan Kankanhalli. 2022. Can Bad Teaching Induce Forgetting? Unlearning in Deep Networks using an Incompetent Teacher. *arXiv preprint arXiv:2205.08096* (2022).

[29] Vikram S Chundawat, Ayush K Tarun, Murari Mandal, and Mohan Kankanhalli. 2022. Zero-shot machine unlearning. *arXiv preprint arXiv:2201.05629* (2022).

[30] Weilin Cong and Mehrdad Mahdavi. 2022. GRAPHEDITOR: An Efficient Graph Representation Learning and Unlearning Approach. *https://congweilin.github.io/CongWeilin.io/* (2022).

[31] Weilin Cong and Mehrdad Mahdavi. 2022. Privacy Matters! Efficient Graph Representation Unlearning with Data Removal Guarantee. *https://congweilin.github.io/CongWeilin.io/* (2022).

[32] Weilin Cong and Mehrdad Mahdavi. 2023. Efficiently forgetting what you have learned in graph representation learning via projection. In *AISTATS*. 6674–6703.

[33] Damai Dai, Li Dong, et al. 2022. Knowledge Neurons in Pretrained Transformers. In *ACL*. 8493–8502.

[34] Quang-Vinh Dang. 2021. Right to Be Forgotten in the Age of Machine Learning. In *ICADS*. 403–411.

[35] Jia Deng, Wei Dong, Richard Socher, et al. 2009. Imagenet: A large-scale hierarchical image database. In *CVPR*. 248–255.

[36] Nicola K Dinsdale, Mark Jenkinson, et al. 2020. Unlearning scanner bias for mri harmonisation. In *MICCAI*. 369–378.

[37] Nicola K Dinsdale, Mark Jenkinson, and Ana IL Namburete. 2021. Deep learning-based unlearning of dataset bias for MRI harmonisation and confound removal. *NeuroImage* 228 (2021), 117689.

[38] Min Du, Zhi Chen, et al. 2019. Lifelong anomaly detection through unlearning. In *SIGSAC*. 1283–1297.

[39] Hua Duan, Hua Li, Guoping He, and Qingtian Zeng. 2007. Decremental learning algorithms for nonlinear langrangian and least squares support vector machines. In *OSB*. 358–366.

[40] Piotr Duda, Maciej Jaworski, Andrzej Cader, and Lipo Wang. 2020. On training deep neural networks using a streaming approach. *JAISCR* 10 (2020).

[41] Yonatan Dukler, Benjamin Bowman, Alessandro Achille, Aditya Golatkar, Ashwin Swaminathan, and Stefano Soatto. 2023. Safe: Machine unlearning with shard graphs. In *ICCV*. 17108–17118.

[42] Cynthia Dwork. 2008. Differential privacy: A survey of results. In *TAMC*. 1–19.

[43] Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* 9, 3–4 (2014), 211–407.

[44] Thorsten Eisenhofer, Doreen Riepel, Varun Chandrasekaran, Esha Ghosh, Olga Ohrimenko, and Nicolas Papernot. 2022. Verifiable and Provably Secure Machine Unlearning. *arXiv preprint arXiv:2210.09126* (2022).

[45] Daniel L Felps, Amelia D Schwickerath, Joyce D Williams, Trung N Vuong, Alan Briggs, Matthew Hunt, Evan Sakmar, David D Saranchak, and Tyler Shumaker. 2020. Class Clown: Data Redaction in Machine Unlearning at Enterprise Scale. *arXiv preprint arXiv:2012.04699* (2020).

[46] Stefan Feuerriegel, Mateusz Dolata, and Gerhard Schwabe. 2020. Fair AI. *Business & information systems engineering* 62, 4 (2020), 379–384.

[47] Yann Fraboni, Martin Van Waerebeke, Kevin Scaman, Richard Vidal, Laetitia Kameni, and Marco Lorenzi. 2024. SIFU: Sequential Informed Federated Unlearning for Efficient and Provable Client Unlearning in Federated Optimization. In *AISTATS*. 3457–3465.

[48] Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. 2014. Privacy in pharmacogenetics: An {End-to-End} case study of personalized warfarin dosing. In *USENIX Security*. 17–32.

[49] Shaopeng Fu, Fengxiang He, et al. 2022. Knowledge Removal in Sampling-based Bayesian Inference. In *ICLR*.

[50] Shaopeng Fu, Fengxiang He, Yue Xu, and Dacheng Tao. 2021. Bayesian inference forgetting. *arXiv preprint arXiv:2101.06417* (2021).

[51] Christian Ganhör, David Penz, Navid Rekabsaz, Oleg Lesota, and Markus Schedl. 2022. Unlearning protected user attributes in recommendations with adversarial training. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2142–2147.

[52] Ji Gao, Sanjam Garg, Mohammad Mahmoody, and Prashant Nalini Vasudevan. 2022. Deletion Inference, Reconstruction, and Compliance in Machine (Un) Learning. *Proc. Priv. Enhancing Technol.* 2022, 3 (2022), 415–436.

[53] Xiangshan Gao, Xingjun Ma, Jingyi Wang, Youcheng Sun, Bo Li, Shouling Ji, Peng Cheng, and Jiming Chen. 2022. VeriFi: Towards Verifiable Federated Unlearning. *arXiv preprint arXiv:2205.12709* (2022).

[54] Sanjam Garg, Shafi Goldwasser, and Prashant Nalini Vasudevan. 2020. Formalizing data deletion in the context of the right to be forgotten. In *EUROCRYPT*. 373–402.

[55] Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Combining Neural Networks with Personalized PageRank for Classification on Graphs. In *ICLR*.

[56] Pierre Geurts, Damien Ernst, et al. 2006. Extremely randomized trees. *Machine learning* 63, 1 (2006), 3–42.

[57] Antonio Ginart, Melody Guan, Gregory Valiant, and James Y Zou. 2019. Making ai forget you: Data deletion in machine learning. *NIPS* 32 (2019).

[58] Shashwat Goel, Ameya Prabhu, and Ponnurangam Kumaraguru. 2022. Evaluating Inexact Unlearning Requires Revisiting Forgetting. *arXiv preprint arXiv:2201.06640* (2022).

[59] Aditya Golatkar, Alessandro Achille, Avinash Ravichandran, Marzia Polito, and Stefano Soatto. 2021. Mixed-privacy forgetting in deep networks. In *CVPR*. 792–801.

[60] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. 2020. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *CVPR*. 9304–9312.

[61] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. 2020. Forgetting outside the box: Scrubbing deep networks of information accessible from input-output observations. In *ECCV*. 383–398.

[62] Adit Goyal, Vikas Hassija, and Victor Hugo C de Albuquerque. 2021. Revisiting Machine Learning Training Process for Enhanced Data Privacy. In *IC3*. 247–251.

[63] Laura Graves, Vineel Nagisetty, and Vijay Ganesh. 2021. Amnesiac machine learning. In *AAAI*, Vol. 35. 11516–11524.

[64] Chuan Guo, Tom Goldstein, Awni Y. Hannun, and Laurens van der Maaten. 2020. Certified Data Removal from Machine Learning Models. In *ICML*, Vol. 119. 3832–3842.

[65] Ruocheng Guo, Lu Cheng, Jundong Li, P Richard Hahn, and Huan Liu. 2020. A survey of learning causality with data: Problems and methods. *CSUR* 53, 4 (2020), 1–37.

[66] Tao Guo, Song Guo, Jiewei Zhang, Wenchao Xu, and Junxiao Wang. 2022. Efficient Attribute Unlearning: Towards Selective Removal of Input Attributes from Feature Representations. *arXiv preprint arXiv:2202.13295* (2022).

[67] Varun Gupta, Christopher Jung, Seth Neel, Aaron Roth, Saeed Sharifi-Malvajerdi, and Chris Waites. 2021. Adaptive machine unlearning. *NIPS* 34 (2021), 16319–16330.

[68] Anisa Halimi, Swanand Kadhe, Ambrish Rawat, and Nathalie Baracaldo. 2022. Federated Unlearning: How to Efficiently Erase a Client in FL? *arXiv preprint arXiv:2207.05521* (2022).

[69] William L Hamilton. 2020. Graph representation learning. *Synthesis Lectures on Artifical Intelligence and Machine Learning* 14, 3 (2020), 1–159.

[70] Johannes Haug and Gjergji Kasneci. 2021. Learning parameter distributions to detect concept drift in data streams. In *ICPR*. 9452–9459.

[71] Katie Hawkins, Nora Alhuwaish, Sana Belguith, Asma Vranaki, and Andrew Charlesworth. 2023. A Decision-Making Process to Implement the 'Right to Be Forgotten'in Machine Learning. In *Annual Privacy Forum*. 20–38.

[72] Yingzhe He, Guozhu Meng, Kai Chen, Jinwen He, and Xingbo Hu. 2021. Deepobliviate: a powerful charm for erasing data residual memory in deep neural networks. *arXiv preprint arXiv:2105.06209* (2021).

[73] Xinshuo Hu, Dongfang Li, Baotian Hu, Zihao Zheng, Zhenyu Liu, and Min Zhang. 2024. Separate the wheat from the chaff: Model deficiency unlearning via parameter-efficient module operation. In *AAAI*, Vol. 38. 18252–18260.

[74] Xinting Hu, Kaihua Tang, Chunyan Miao, Xian-Sheng Hua, and Hanwang Zhang. 2021. Distilling causal effect of data in class-incremental learning. In *CVPR*. 3957–3966.

[75] Hanxun Huang, Xingjun Ma, Sarah Monazam Erfani, James Bailey, and Yisen Wang. 2021. Unlearnable Examples: Making Personal Data Unexploitable. In *ICLR*.

[76] Yangsibo Huang, Xiaoxiao Li, et al. 2021. EMA: Auditing Data Removal from Trained Models. In *MICCAI*. 793–803.

[77] Eyke Hüllermeier and Willem Waegeman. 2021. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning* 110, 3 (2021), 457–506.

[78] Thanh Trung Huynh, Trong Bang Nguyen, Phi Le Nguyen, Thanh Tam Nguyen, Matthias Weidlich, Quoc Viet Hung Nguyen, and Karl Aberer. 2024. Fast-fedul: A training-free federated unlearning with provable skew resilience. In *ECML-PKDD*. 55–72.

[79] Zachary Izzo, Mary Anne Smart, Kamalika Chaudhuri, and James Zou. 2021. Approximate data deletion from machine learning models. In *AISTAT*. 2008–2016.

[80] Matthew Jagielski, Om Thakkar, Florian Tramèr, Daphne Ippolito, Katherine Lee, Nicholas Carlini, Eric Wallace, Shuang Song, Abhradeep Thakurta, Nicolas Papernot, et al. 2022. Measuring Forgetting of Memorized Training Examples. *arXiv preprint arXiv:2207.00099* (2022).

[81] Hengrui Jia, Mohammad Yaghini, Christopher A Choquette-Choo, Natalie Dullerud, Anvith Thudi, Varun Chandrasekaran, and Nicolas Papernot. 2021. Proof-of-learning: Definitions and practice. In *SP*. 1039–1056.

[82] Sharu Theresa Jose and Osvaldo Simeone. 2021. A unified PAC-Bayesian framework for machine unlearning via information risk minimization. In *MLSP*. 1–6.

[83] Swanand Kadhe, Anisa Halimi, Ambrish Rawat, and Nathalie Baracaldo. 2023. FairSISA: Ensemble Post-Processing to Improve Fairness of Unlearning in LLMs. In *SoLaR*.

[84] Masayuki Karasuyama and Ichiro Takeuchi. 2009. Multiple incremental decremental learning of support vector machines. *NIPS* 22 (2009).

[85] Masayuki Karasuyama and Ichiro Takeuchi. 2010. Multiple incremental decremental learning of support vector machines. *IEEE Transactions on Neural Networks* 21, 7 (2010), 1048–1059.

[86] Aly Kassem, Omar Mahmoud, and Sherif Saad. 2023. Preserving privacy through dememorization: An unlearning technique for mitigating memorization risks in language models. In *EMNLP*. 4360–4379.

[87] Michael Kearns. 1998. Efficient noise-tolerant learning from statistical queries. *JACM* 45, 6 (1998), 983–1006.

[88] Mohammad Emtiyaz E Khan et al. 2021. Knowledge-adaptation priors. *NIPS* 34 (2021), 19757–19770.

[89] Hyunjune Kim, Sangyong Lee, and Simon S Woo. 2024. Layer Attack Unlearning: Fast and Accurate Machine Unlearning via Layer Level Attack and Knowledge Distillation. In *AAAI*, Vol. 38. 21241–21248.

[90] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *PNAS* 114, 13 (2017), 3521–3526.

[91] Pang Wei Koh et al. 2017. Understanding black-box predictions via influence functions. In *ICML*. 1885–1894.

[92] S. Kullback et al. 1951. On Information and Sufficiency. *The Annals of Mathematical Statistics* 22, 1 (1951), 79 – 86.

[93] Sangyoon Lee and Dae-Hyun Choi. 2024. Learning and Unlearning to Operate Profitable Secure Electric Vehicle Charging. *TII* (2024).

[94] Sang-Woong Lee, Jawad Tanveer, Amir Masoud Rahmani, Hamid Alinejad-Rokny, Parisa Khoshvaght, Gholamreza Zare, Pegah Malekpour Alamdari, and Mehdi Hosseinzadeh. 2024. SFGCN: Synergetic Fusion-based Graph Convolutional Networks Approach for Link Prediction in Social Networks. *Information Fusion* (2024), 102684.

[95] Na Lei, Kehua Su, Li Cui, Shing-Tung Yau, and Xianfeng David Gu. 2019. A geometric view of optimal transportation and generative model. *Computer Aided Geometric Design* 68 (2019), 1–21.

[96] Guihong Li, Hsiang Hsu, Chun-Fu Chen, and Radu Marculescu. 2024. Machine Unlearning for Image-to-Image Generative Models. In *ICLR*.

[97] Xunkai Li, Yulin Zhao, Zhengyu Wu, Wentao Zhang, Rong-Hua Li, and Guoren Wang. 2024. Towards Effective and General Graph Unlearning via Mutual Evolution. In *AAAI*, Vol. 38. 13682–13690.

[98] Yuyuan Li, Chaochao Chen, Xiaolin Zheng, Yizhao Zhang, Zhongxuan Han, Dan Meng, and Jun Wang. 2023. Making users indistinguishable: Attribute-wise unlearning in recommender systems. In *MM*. 984–994.

[99] Yuantong Li, Chi-Hua Wang, and Guang Cheng. 2021. Online Forgetting Process for Linear Regression Models. In *AISTAT*, Vol. 130. 217–225.

[100] Shen Lin, Xiaoyu Zhang, Chenyang Chen, Xiaofeng Chen, and Willy Susilo. 2023. Erm-ktp: Knowledge-level machine unlearning via knowledge transfer. In *CVPR*. 20147–20155.

[101] Bo Liu, Qiang Liu, et al. 2022. Continual Learning and Private Unlearning. *arXiv preprint arXiv:2203.12817* (2022).

[102] Gaoyang Liu, Xiaoqiang Ma, Yang Yang, Chen Wang, and Jiangchuan Liu. 2020. Federated unlearning. *arXiv preprint arXiv:2012.13891* (2020).

[103] Gaoyang Liu, Xiaoqiang Ma, Yang Yang, Chen Wang, and Jiangchuan Liu. 2021. Federaser: Enabling efficient client-level data removal from federated learning models. In *IWQOS*. 1–10.

[104] Wenyan Liu, Juncheng Wan, Xiaoling Wang, Weinan Zhang, Dell Zhang, and Hang Li. 2022. Forgetting fast in recommender systems. *arXiv preprint arXiv:2208.06875* (2022).

[105] Xiao Liu and Sotirios A Tsaftaris. 2020. Have you forgotten? A method to assess if machine learning models have forgotten data. In *MICCAI*. 95–105.

[106] Yang Liu, Mingyuan Fan, Cen Chen, Ximeng Liu, Zhuo Ma, Li Wang, and Jianfeng Ma. 2022. Backdoor Defense with Machine Unlearning. *arXiv preprint arXiv:2201.09538* (2022).

[107] Yang Liu, Zhuo Ma, Ximeng Liu, Jian Liu, Zhongyuan Jiang, Jianfeng Ma, Philip Yu, and Kui Ren. 2020. Learn to forget: Machine unlearning via neuron masking. *arXiv preprint arXiv:2003.10933* (2020).

[108] Yang Liu, Zhuo Ma, Yilong Yang, Ximeng Liu, Jianfeng Ma, and Kui Ren. 2021. Revfrf: Enabling cross-domain random forest training with revocable federated learning. *TDSC* (2021).

[109] Yi Liu, Lei Xu, Xingliang Yuan, Cong Wang, and Bo Li. 2022. The Right to be Forgotten in Federated Learning: An Efficient Realization with Rapid Retraining. In *INFOCOM*. 1749–1758.

[110] Ziyao Liu, Yu Jiang, Jiyuan Shen, Minyi Peng, Kwok-Yan Lam, Xingliang Yuan, and Xiaoning Liu. 2023. A survey on federated unlearning: Challenges, methods, and future directions. *CSUR* (2023).

[111] Serveh Lotfi, Mitra Mirzarezaee, Mehdi Hosseinzadeh, and Vahid Seydi. 2021. Detection of rumor conversations in Twitter using graph convolutional networks. *Applied Intelligence* 51 (2021), 4774–4787.

[112] Ximing Lu, Sean Welleck, Jack Hessel, Liwei Jiang, Lianhui Qin, Peter West, Prithviraj Ammanabrolu, and Yejin Choi. 2022. Quark: Controllable text generation with reinforced unlearning. *NeurIPS* 35 (2022), 27591–27609.

[113] Ananth Mahadevan and Michael Mathioudakis. 2021. Certifiable machine unlearning for linear models. *arXiv preprint arXiv:2106.15093* (2021).

[114] Ananth Mahadevan and Michael Mathioudakis. 2022. Certifiable Unlearning Pipelines for Logistic Regression: An Experimental Study. *Machine Learning and Knowledge Extraction* 4, 3 (2022), 591–620.

[115] Alessandro Mantelero. 2013. The EU Proposal for a General Data Protection Regulation and the roots of the 'right to be forgotten'. *Computer Law & Security Review* 29, 3 (2013), 229–235.

[116] Neil G Marchant, Benjamin IP Rubinstein, and Scott Alfeld. 2022. Hard to Forget: Poisoning Attacks on Certified Machine Unlearning. In *AAAI*, Vol. 36. 7691–7700.

[117] James Martens. 2020. New insights and perspectives on the natural gradient method. *JMLR* 21, 1 (2020), 5776–5851.

[118] Iacopo Masi, Yue Wu, Tal Hassner, and Prem Natarajan. 2018. Deep face recognition: A survey. In *SIBGRAPI*. 471–478.

[119] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *AISTAT*. 1273–1282.

[120] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2021. A survey on bias and fairness in machine learning. *CSUR* 54, 6 (2021), 1–35.

[121] Ronak Mehta, Sourav Pal, Vikas Singh, and Sathya N Ravi. 2022. Deep Unlearning via Randomized Conditionally Independent Hessians. In *CVPR*. 10422–10431.

[122] Salvatore Mercuri, Raad Khraishi, Ramin Okhrati, Devesh Batra, Conor Hamill, Taha Ghasempour, and Andrew Nowlan. 2022. An Introduction to Machine Unlearning. *arXiv preprint arXiv:2209.00939* (2022).

[123] Salvatore Mercuri, Raad Khraishi, Ramin Okhrati, Devesh Batra, Conor Hamill, Taha Ghasempour, and Andrew Nowlan. 2022. An introduction to machine unlearning. *arXiv preprint arXiv:2209.00939* (2022).

[124] Paul Micaelli et al. 2019. Zero-shot knowledge transfer via adversarial belief matching. *NIPS* 32 (2019).

[125] Junhyun Nam, Hyuntak Cha, Sungsoo Ahn, Jaeho Lee, and Jinwoo Shin. 2020. Learning from failure: De-biasing classifier from biased classifier. *NIPS* 33 (2020), 20673–20684.

[126] Seth Neel, Aaron Roth, and Saeed Sharifi-Malvajerdi. 2021. Descent-to-delete: Gradient-based methods for machine unlearning. In *Algorithmic Learning Theory*. 931–962.

[127] Quoc Phong Nguyen, Bryan Kian Hsiang Low, and Patrick Jaillet. 2020. Variational bayesian unlearning. *NIPS* 33 (2020), 16025–16036.

[128] Quoc Phong Nguyen, Ryutaro Oikawa, Dinil Mon Divakaran, Mun Choon Chan, et al. 2022. Markov Chain Monte Carlo-Based Machine Unlearning: Unlearning What Needs to Be Forgotten. In *ASIACCS*. 351–363.

[129] Thanh Tam Nguyen. 2019. *Debunking Misinformation on the Web: Detection, Validation, and Visualisation.* Ph.D. Dissertation. EPFL, Switzerland.

[130] Thanh Tam Nguyen, Thanh Trung Huynh, Phi Le Nguyen, Alan Wee-Chung Liew, Hongzhi Yin, and Quoc Viet Hung Nguyen. 2022. A Survey of Machine Unlearning. *arXiv preprint arXiv:2209.02299* (2022).

[131] Thanh Toan Nguyen, Thanh Tam Nguyen, Thanh Thi Nguyen, Bay Vo, Jun Jo, and Quoc Viet Hung Nguyen. 2021. Judo: Just-in-time rumour detection in streaming social platforms. *Information Sciences* 570 (2021), 70–93.

[132] Chao Pan, Eli Chien, and Olgica Milenkovic. 2023. Unlearning graph classifiers with limited data resources. In *WWW*. 716–726.

[133] Subhodip Panda and Prathosh AP. 2023. FAST: Feature Aware Similarity Thresholding for Weak Unlearning in Black-Box Generative Models. *arXiv preprint arXiv:2312.14895* (2023).

[134] Stuart L Pardau. 2018. The California consumer privacy act: Towards a European-style privacy regime in the United States. *J. Tech. L. & Pol'y* 23 (2018), 68.

[135] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. 2019. Continual lifelong learning with neural networks: A review. *Neural Networks* 113 (2019), 54–71.

[136] Nishchal Parne, Kyathi Puppaala, Nithish Bhupathi, and Ripon Patgiri. 2021. An Investigation on Learning, Polluting, and Unlearning the Spam Emails for Lifelong Learning. *arXiv preprint arXiv:2111.14609* (2021).

[137] Tim Pearce, Felix Leibfried, and Alexandra Brintrup. 2020. Uncertainty in neural networks: Approximately bayesian ensembling. In *AISTATS*. 234–244.

[138] Alexandra Peste, Dan Alistarh, and Christoph H Lampert. 2021. SSSE: Efficiently Erasing Samples from Trained Machine Learning Models. In *NeurIPS 2021 Workshop Privacy in Machine Learning*.

[139] Nicholas Pochinkov and Nandi Schoots. 2024. Dissecting Language Models: Machine Unlearning via Selective Pruning. *arXiv preprint arXiv:2403.01267* (2024).

[140] Hongyu Qiu, Yongwei Wang, Yonghui Xu, Lizhen Cui, and Zhiqi Shen. 2023. FedCIO: Efficient Exact Federated Unlearning with Clustering, Isolation, and One-shot Aggregation. In *BigData*. 5559–5568.

[141] Vikram V Ramaswamy, Sunnie SY Kim, and Olga Russakovsky. 2021. Fair attribute classification through latent space de-biasing. In *CVPR*. 9301–9310.

[142] Kui Ren, Tianhang Zheng, Zhan Qin, and Xue Liu. 2020. Adversarial attacks and defenses in deep learning. *Engineering* 6, 3 (2020), 346–360.

[143] Zhao Ren, Alice Baird, Jing Han, Zixing Zhang, and Björn Schuller. 2020. Generating and protecting against adversarial attacks for deep speech-based emotion recognition models. In *ICASSP*. 7184–7188.

[144] Zhao Ren, Jing Han, Nicholas Cummins, and Björn W Schuller. 2020. Enhancing Transferability of Black-Box Adversarial Attacks via Lifelong Learning for Speech Emotion Recognition Models.. In *INTERSPEECH*. 496–500.

[145] Zhao Ren, Thanh Tam Nguyen, and Wolfgang Nejdl. 2022. Prototype learning for interpretable respiratory sound analysis. In *ICASSP*. 9087–9091.

[146] Enrique Romero, Ignacio Barrio, and Lluís Belanche. 2007. Incremental and decremental learning for linear support vector machines. In *ICANN*. 209–218.

[147] Wolfgang Roth and Franz Pernkopf. 2018. Bayesian neural networks with weight sharing using Dirichlet processes. *TPAMI* 42, 1 (2018), 246–252.

[148] Ahmed Salem, Apratim Bhattacharya, Michael Backes, Mario Fritz, and Yang Zhang. 2020. {Updates-Leak}: Data Set Inference and Reconstruction Attacks in Online Learning. In *USENIX Security*. 1291–1308.

[149] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. 2019. ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models. In *NDSS*.

[150] WN Sari, BS Samosir, N Sahara, L Agustina, and Y Anita. 2020. Learning mathematics "Asyik" with Youtube educative media. In *Journal of Physics: Conference Series*, Vol. 1477. 022012.

[151] Felix Sattler, Tim Korjakow, Roman Rischke, and Wojciech Samek. 2021. Fedaux: Leveraging unlabeled auxiliary data in federated learning. *TNNLS* (2021).

[152] Sebastian Schelter. 2020. "Amnesia" - A Selection of Machine Learning Models That Can Forget User Data Very Fast. In *CIDR*.

[153] Sebastian Schelter, Stefan Grafberger, and Ted Dunning. 2021. Hedgecut: Maintaining randomised trees for low-latency machine unlearning. In *SIGMOD*. 1545–1557.

[154] Ayush Sekhari, Jayadev Acharya, Gautam Kamath, and Ananda Theertha Suresh. 2021. Remember what you want to forget: Algorithms for machine unlearning. *NIPS* 34 (2021), 18075–18086.

[155] Thanveer Shaik, Xiaohui Tao, Lin Li, Haoran Xie, Taotao Cai, Xiaofeng Zhu, and Qing Li. 2024. Framu: Attention-based machine unlearning using federated reinforcement learning. *TKDE* (2024).

[156] Thanveer Shaik, Xiaohui Tao, Haoran Xie, Lin Li, Xiaofeng Zhu, and Qing Li. 2023. Exploring the landscape of machine unlearning: A comprehensive survey and taxonomy. *arXiv preprint arXiv:2305.06360* (2023).

[157] S Shan, E Wenger, J Zhang, H Li, H Zheng, and BY Zhao. 2020. Protecting Personal Privacy against Una uthorized Deep Learning Models. In *USENIX Security*. 1–16.

[158] Jiaqi Shao, Tao Lin, Xuanyu Cao, and Bing Luo. 2024. Federated Unlearning: a Perspective of Stability and Fairness. *arXiv preprint arXiv:2402.01276* (2024).

[159] Takashi Shibata, Go Irie, Daiki Ikami, and Yu Mitsuzumi. 2021. Learning with Selective Forgetting.. In *IJCAI*, Vol. 2. 6.

[160] Saurabh Shintre et al. 2019. Making machine learning forget. In *Annual Privacy Forum*. 72–83.

[161] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *SP*. 3–18.

[162] Ravid Shwartz-Ziv and Naftali Tishby. 2017. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810* (2017).

[163] Abhijeet Singh and Abhineet Anand. 2017. Data leakage detection using cloud computing. *IJECS* 6, 4 (2017).

[164] Richa Singh, Puspita Majumdar, Surbhi Mittal, and Mayank Vatsa. 2022. Anatomizing bias in facial analysis. In *AAAI*, Vol. 36. 12351–12358.

[165] Yash Sinha, Murari Mandal, and Mohan Kankanhalli. 2024. Multi-Modal Recommendation Unlearning. *arXiv preprint arXiv:2405.15328* (2024).

[166] David Marco Sommer, Liwei Song, Sameer Wagh, and Prateek Mittal. 2020. Towards probabilistic verification of machine unlearning. *arXiv preprint arXiv:2003.04247* (2020).

[167] David Marco Sommer, Liwei Song, Sameer Wagh, and Prateek Mittal. 2022. Athena: Probabilistic Verification of Machine Unlearning. *Proc. Priv. Enhancing Technol.* 2022, 3 (2022), 268–290.

[168] Aman Tahiliani, Vikas Hassija, Vinay Chamola, and Mohsen Guizani. 2021. Machine Unlearning: Its Need and Implementation Strategies. In *IC3*. 241–246.

[169] Nguyen Thanh Tam, Matthias Weidlich, Duong Chi Thang, Hongzhi Yin, and Nguyen Quoc Viet Hung. 2017. Retaining Data from Streams of Social Platforms with Minimal Regret. In *IJCAI*. 2850–2856.

[170] Jafar Tanha, Yousef Abdi, Negin Samadi, Nazila Razzaghi, and Mohammad Asadpour. 2020. Boosting methods for multi-class imbalanced data classification: an experimental review. *Journal of Big Data* 7, 1 (2020), 1–47.

[171] Youming Tao, Cheng-Long Wang, Miao Pan, Dongxiao Yu, Xiuzhen Cheng, and Di Wang. 2024. Communication Efficient and Provable Federated Unlearning. *PVLDB* 17, 5 (2024), 1119–1131.

[172] Ayush K Tarun, Vikram S Chundawat, Murari Mandal, and Mohan Kankanhalli. 2021. Fast yet effective machine unlearning. *arXiv preprint arXiv:2111.08947* (2021).

[173] Anvith Thudi, Gabriel Deza, Varun Chandrasekaran, and Nicolas Papernot. 2022. Unrolling sgd: Understanding factors influencing machine unlearning. In *EuroS&P*. 303–319.

[174] Anvith Thudi, Hengrui Jia, Ilia Shumailov, and Nicolas Papernot. 2022. On the necessity of auditable algorithmic definitions for machine unlearning. In *USENIX Security*. 4007–4022.

[175] Anvith Thudi, Ilia Shumailov, Franziska Boenisch, and Nicolas Papernot. 2022. Bounding Membership Inference. *arXiv preprint arXiv:2202.12232* (2022).

[176] Naftali Tishby et al. 2000. The information bottleneck method. *arXiv preprint physics/0004057* (2000).

[177] Naftali Tishby and Noga Zaslavsky. 2015. Deep learning and the information bottleneck principle. In *ITW*. 1–5.

[178] Piyush Tiwary, Atri Guha, Subhodip Panda, et al. 2023. Adapt then unlearn: Exploiting parameter space semantics for unlearning in generative adversarial networks. *arXiv preprint arXiv:2309.14054* (2023).

[179] Cheng-Hao Tsai, Chieh-Yen Lin, and Chih-Jen Lin. 2014. Incremental and decremental training for linear classification. In *KDD*. 343–352.

[180] Amund Tveit et al. 2003. Multicategory incremental proximal support vector classifiers. In *KES*. 386–392.

[181] Amund Tveit, Magnus Lie Hetland, and Håavard Engum. 2003. Incremental and decremental proximal support vector classification using decay coefficients. In *DaWaK*. 422–429.

[182] Enayat Ullah, Tung Mai, Anup Rao, Ryan A Rossi, and Raman Arora. 2021. Machine unlearning via algorithmic stability. In *Conference on Learning Theory*. 4126–4142.

[183] Michael Veale, Reuben Binns, and Lilian Edwards. 2018. Algorithms that remember: model inversion attacks and data protection law. *Philos. Trans. R. Soc. A* 376, 2133 (2018), 20180083.

[184] Sergio Verdú. 2014. Total variation distance and the distribution of relative information. In *ITA*. 1–3.

[185] Eduard Fosch Villaronga, Peter Kieseberg, and Tiffany Li. 2018. Humans forget, machines remember: Artificial intelligence and the right to be forgotten. *Computer Law & Security Review* 34, 2 (2018), 304–313.

[186] Paul Voigt and Axel Von dem Bussche. 2017. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing* 10, 3152676 (2017), 10–5555.

[187] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. 2019. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *SP*. 707–723.

[188] Benjamin Longxiang Wang and Sebastian Schelter. 2022. Efficiently Maintaining Next Basket Recommendations under Additions and Deletions of Baskets and Items. *arXiv preprint arXiv:2201.13313* (2022).

[189] Cheng-Long Wang, Mengdi Huai, and Di Wang. 2023. Inductive graph unlearning. In *USENIX*. 3205–3222.

[190] Hangyu Wang, Jianghao Lin, Bo Chen, Yang Yang, Ruiming Tang, Weinan Zhang, and Yong Yu. 2024. Towards Efficient and Effective Unlearning of Large Language Models for Recommendation. *arXiv preprint arXiv:2403.03536* (2024).

[191] Junxiao Wang, Song Guo, et al. 2022. Federated unlearning via class-discriminative pruning. In *WWW*. 622–632.

[192] Lingzhi Wang, Tong Chen, Wei Yuan, Xingshan Zeng, Kam-Fai Wong, and Hongzhi Yin. 2023. KGA: A General Machine Unlearning Framework Based on Knowledge Gap Alignment. In *ACL*. 13264–13276.

[193] Qizhou Wang, Bo Han, Puning Yang, Jianing Zhu, Tongliang Liu, and Masashi Sugiyama. 2024. Unlearning with Control: Assessing Real-world Utility for Large Language Model Unlearning. *arXiv preprint arXiv:2406.09179* (2024).

[194] Rui Wang, Yong Fuga Li, XiaoFeng Wang, Haixu Tang, and Xiaoyong Zhou. 2009. Learning your identity and disease from research papers: information leaks in genome wide association study. In *CCS*. 534–544.

[195] Zeyu Wang, Klint Qinami, Ioannis Christos Karakozis, Kyle Genova, Prem Nair, Kenji Hata, and Olga Russakovsky. 2020. Towards fairness in visual recognition: Effective strategies for bias mitigation. In *CVPR*. 8919–8928.

[196] Alexander Warnecke, Lukas Pirch, Christian Wressnegger, and Konrad Rieck. 2021. Machine Unlearning of Features and Labels. *arXiv preprint arXiv:2108.11577* (2021).

[197] Chen Wu, Sencun Zhu, and Prasenjit Mitra. 2022. Federated unlearning with knowledge distillation. *arXiv preprint arXiv:2201.09441* (2022).

[198] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *ICML*. 6861–6871.

[199] Ga Wu, Masoud Hashemi, and Christopher Srinivasa. 2022. PUMA: Performance Unchanged Model Augmentation for Training Data Removal. In *AAAI*.

[200] Leijie Wu, Song Guo, Junxiao Wang, Zicong Hong, Jie Zhang, and Yaohong Ding. 2022. Federated Unlearning: Guarantee the Right of Clients to Forget. *IEEE Network* 36, 5 (2022), 129–135.

[201] Yinjun Wu et al. 2020. Deltagrad: Rapid retraining of machine learning models. In *ICML*. 10355–10366.

[202] Yinjun Wu, Val Tannen, and Susan B Davidson. 2020. Priu: A provenance-based approach for incrementally updating regression models. In *SIGMOD*. 447–462.

[203] Zhaomin Wu, Junhui Zhu, Qinbin Li, and Bingsheng He. 2023. Deltaboost: Gradient boosting decision trees with efficient machine unlearning. *SIGMOD* 1, 2 (2023), 1–26.

[204] Heng Xu, Tianqing Zhu, Lefeng Zhang, Wanlei Zhou, and Philip S. Yu. 2023. Machine Unlearning: A Survey. *CSUR* 56, 1 (2023), 36.

[205] Jie Xu, Zihan Wu, Cong Wang, and Xiaohua Jia. 2024. Machine unlearning: Solutions and challenges. *TETCI* (2024).

[206] Tomoya Yamashita, Masanori Yamada, and Takashi Shibata. 2023. One-Shot Machine Unlearning with Mnemonic Code. *arXiv preprint arXiv:2306.05670* (2023).

[207] Youngsik Yoon, Jinhwan Nam, Hyojeong Yun, Dongwoo Kim, and Jungseul Ok. 2022. Few-Shot Unlearning by Model Inversion. *arXiv preprint arXiv:2205.15567* (2022).

[208] Da Yu, Huishuai Zhang, Wei Chen, Jian Yin, and Tie-Yan Liu. 2021. How does data augmentation affect privacy in machine learning?. In *AAAI*, Vol. 35. 10746–10753.

[209] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. 2015. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365* (2015).

[210] Wei Yuan, Hongzhi Yin, Fangzhao Wu, Shijie Zhang, Tieke He, and Hao Wang. 2023. Federated unlearning for on-device recommendation. In *WSDM*. 393–401.

[211] Santiago Zanella-Béguelin, Lukas Wutschitz, Shruti Tople, Victor Rühle, Andrew Paverd, Olga Ohrimenko, et al. 2020. Analyzing information leakage of updates to natural language models. In *SIGSAC*. 363–375.

[212] Yingyan Zeng, Tianhao Wang, Si Chen, Hoang Anh Just, Ran Jin, and Ruoxi Jia. 2021. Learning to Refit for Convex Learning Problems. *arXiv preprint arXiv:2111.12545* (2021).

[213] Hao Zhang, Bo Chen, Yulai Cong, Dandan Guo, Hongwei Liu, and Mingyuan Zhou. 2020. Deep autoencoding topic model with scalable hybrid Bayesian inference. *TPAMI* 43, 12 (2020), 4306–4322.

[214] Peng-Fei Zhang, Guangdong Bai, Zi Huang, and Xin-Shun Xu. 2022. Machine Unlearning for Image Retrieval: A Generative Scrubbing Approach. In *MM*. 237–245.

[215] Xulong Zhang, Jianzong Wang, Ning Cheng, Yifu Sun, Chuanyao Zhang, and Jing Xiao. 2023. Machine unlearning methodology based on stochastic teacher network. In *ADMA*. 250–261.

[216] Yimeng Zhang, Xin Chen, Jinghan Jia, Yihua Zhang, Chongyu Fan, Jiancheng Liu, Mingyi Hong, Ke Ding, and Sijia Liu. 2024. Defensive Unlearning with Adversarial Training for Robust Concept Erasure in Diffusion Models. *arXiv preprint arXiv:2405.15234* (2024).

[217] Yihua Zhang, Yimeng Zhang, Yuguang Yao, Jinghan Jia, Jiancheng Liu, Xiaoming Liu, and Sijia Liu. 2024. Unlearncanvas: A stylized image dataset to benchmark machine unlearning for diffusion models. *arXiv preprint arXiv:2402.11846* (2024).

[218] Jianing Zhu, Bo Han, Jiangchao Yao, Jianliang Xu, Gang Niu, and Masashi Sugiyama. 2024. Decoupling the Class Label and the Target Concept in Machine Unlearning. *arXiv preprint arXiv:2406.08288* (2024).

[219] Xiangrong Zhu, Guangyao Li, and Wei Hu. 2023. Heterogeneous federated knowledge graph embedding learning and unlearning. In *WWW*. 2444–2454.

[220] James Zou and Londa Schiebinger. 2018. AI can be sexist and racist – it's time to make it fair.