




Full Length Article

PRUNE: A patching based repair framework for certifiable and privacy-robust unlearning of neural networks

Xuran Li ^a, Jingyi Wang ^{a,b,*}, Xiaohan Yuan ^a, Peixin Zhang ^c

^a Zhejiang University, Hangzhou, Zhejiang, 310027, China

^b Huzhou Institute of Industrial Control Technology, Huzhou, Zhejiang, 313000, China

^c Singapore Management University, Singapore, 188065, Singapore

ARTICLE INFO

Keywords:

Machine learning
Machine unlearning
Privacy leakage
Data privacy

ABSTRACT

Machine unlearning has emerged as a key mechanism for enabling the “right to be forgotten” in neural network models, allowing the selective removal of specific training data upon request. Existing approaches typically rely on retraining models with the remaining data, which is computationally expensive and difficult to verify, especially when deployed models are distributed or resource-constrained. To address this challenge, our prior conference work introduced PRUNE, a patching-based framework that formulates unlearning as a neural network repair problem. PRUNE achieves targeted forgetting by learning lightweight patch networks that redirect model predictions on the data to be unlearned while preserving performance on the remaining data. In this extended journal version, we make three major advances: (1) we formally define a threat model that characterizes dishonest behaviors of model owners and corresponding privacy risks; (2) we extend PRUNE to support class-level unlearning, enabling removal of all samples from a target category; and (3) we perform additional experiments showing that PRUNE can resist membership inference attacks, demonstrating its privacy robustness. Extensive evaluations on multiple classification benchmarks confirm that PRUNE achieves certifiable unlearning with high efficiency, minimal performance degradation, and strong verifiability.

1. Introduction

In this data-driven era, people’s personal data are inevitably used on a large scale. Many countries and regions have introduced relevant regulations or laws on how to use these data properly and protect people’s various rights. One representative regulation, GDPR (European Union, 2016), stipulates that users have the *right to be forgotten*. When this right is applied in machine learning (ML), the user should be able to make a request to withdraw their data used for training a model (e.g., a neural network), which the model owner should execute (Ginart et al., 2019).

To support the request of data erasure (or removal), the study on *machine unlearning* (Bourtoule et al., 2021; Chen et al., 2022b; Foster et al., 2024; Guo et al., 2020; Nguyen et al., 2020; Shibata et al., 2021; Yan et al., 2022; Zhou et al., 2025) has emerged. Arguably, the most intuitive way to unlearn is to retrain an alternative model M_r after removing the data to be withdrawn from the training set¹. Existing unlearning methods are mostly measured by the distance between the unlearned model M_U and M_r , and can be roughly divided into two categories: *exact unlearning* and *approximate unlearning*. The general idea of exact unlearning is

to retrain a model (without the data to remove) using different speedup approaches (Bourtoule et al., 2021; Chen et al., 2022a,b; Hu et al., 2024; Li et al., 2021; Liu et al., 2022). They often perform additional operations during the model training phase to reduce the cost of retraining by, for example, either slicing the data (Bourtoule et al., 2021) or segmenting the training (Yan et al., 2022). Approximate unlearning (Chien et al., 2023; Foster et al., 2024; Ginart et al., 2019; Guo et al., 2020; Izzo et al., 2021; Thudi et al., 2022a; Warnecke et al., 2023; Zhang et al., 2022), on the other hand, aims to approximate the performance of M_r by modifying the model parameters to save the retraining cost. For example, the influence function is used to estimate the impact of data withdrawal on the model, and the model parameters are updated on this basis (Warnecke et al., 2023; Wu et al., 2022, 2023b). Such approaches typically rely on global or distributed parameter updates, which may unintentionally affect the model’s behavior on data unrelated to the removal request.

However, as pointed out by Thudi et al. (2022b), one limitation of these unlearning recipes is that *it remains difficult to erase the data holder’s privacy concern or convince a third-party auditor as the retrained model may*

* Corresponding author.

E-mail address: wangjyee@zju.edu.cn (J. Wang).

¹ We use M_r to denote the retrained model without the data to remove and M_U the model obtained after different kinds of unlearning methods consistently.

still perform well on an unpredictable portion (even most) of the data to unlearn. This is especially the case for real-world scenarios when a large number of data holders jointly train a high-performance model like a neural network (NN) where the data can be overlapping to a significant degree. What's worse, this limitation may further be exploited by the model owner to trick the data holders into believing that they have executed the unlearning operation but actually not, given the challenge in auditing unlearning (Gao et al., 2022; Zhou et al., 2023). This is highly attractive for the model owner since machine unlearning inevitably adds additional computational cost, bringing in a natural conflict of interest between the model owner and the data holders who are making data withdrawal requests. In a more practical setting, we simply cannot assume that the model owner is completely honest in the interactive unlearning process. Moreover, retraining is often deemed as a last-minute solution from the honest model owner's perspective due to its high computational cost in general.

Arguably, an ideal unlearning solution should respect the interests of both the data holders and the model owner. To ease the data holder's privacy concern, the model after the unlearning operation is expected to lose its predictive ability on the data to unlearn to an assured degree. From the model owner's perspective, the unlearning operation should be lightweight and its execution should not affect the model's performance on the remaining data. Several more recent studies (Chen et al., 2023; Tarun et al., 2023) have been conducted with such two sides of the coin in mind. But their granularity is too coarse-grained, as they focus on label-level, i.e., forgetting an entire category of data in a classification task. This is a reasonable operation in some scenarios such as face recognition, but for most classification tasks, a few data withdrawal requests are often not supposed enough to affect the entire category. It is thus unacceptable for the model owner if the model loses prediction accuracy for the entire category's data just for a partial delete request.

In this work, we take a new angle and propose a Patching based Repair framework for certifiable and privacy-robust UNlearning (PRUNE) which directly targets addressing two concerning variables simultaneously: 1) the model's performance on the data to remove, and 2) the model's performance on the remaining data. These two variables naturally coincide with the neural network repair problem (Dong et al., 2021; Ma et al., 2024; Sohn et al., 2022; Sotoudeh & Thakur, 2021). The goal of repair is to correct a model's behavior on a small set of identified inputs through targeted and localized modifications, while explicitly constraining unintended side effects on the rest of the input space. Unlike conventional fine-tuning or parameter-update-based approaches that typically involve global optimization over large parameter subsets, neural network repair focuses on minimal, input-selective interventions that preserve the original model parameters as much as possible. This motivates us to connect machine unlearning with neural network repair by drawing an analogy between the data for removal in unlearning and the erroneous data in neural network repair. By doing so, we formulate the unlearning problem as a neural network repair problem, where a similar operation with the opposite objective function can be performed to satisfy the needs of both the data holder and the model owner. Specifically, we first propose to carefully craft a minimum 'patch' network for unlearning a targeted given data point by redirecting the model's prediction elsewhere in a certifiable way. In contrast to fine-tuning, which modifies the original parameters, the patch operates as an external corrective component, leaving the backbone model intact and enabling explicit control over where and how the model behavior is altered. Furthermore, to address more practical scenarios in unlearning a considerable amount of data points (or an entire class), we propose to iteratively select only a small subset of representative data points to unlearn, which however achieves the effect of unlearning the whole set. We extensively evaluated the effectiveness of our approach on multiple categorical datasets. The results show that our approach can achieve easily measurable unlearning while retaining the model's original performance on the remaining data. Besides, our approach is competitive

in terms of efficiency and memory consumption in comparison with various baseline unlearning methods.

This journal version is an extended work of our previous conference paper (Li et al., 2025). In this extension, we make three major improvements: (1) we formally define a threat model to characterize potential dishonest behaviors of model owners and the corresponding privacy risks; (2) we extend our unlearning mechanism to support class-level unlearning, enabling the removal of all samples belonging to a given category; and (3) we conduct additional experiments to verify that our proposed approach can effectively confuse membership inference attacks, demonstrating the robustness of the unlearned model from a privacy perspective. None of this content appeared in the conference version.

In summary, we make the following main contributions:

- We take a new angle and propose a novel unlearning approach that meets the needs of both data holders and the model owner. The approach connects unlearning with neural network repair to directly falsify the model's prediction on the withdrawn data by using a carefully crafted patch network. This allows the data holders to make an intuitive assessment of the model's forgetting effect. To mitigate the impact on the model's performance on the remaining data, the minimality of the patch network is theoretically guaranteed together with the localization (satisfying the interest of the model owners).
- To further cope with more large-scale unlearning scenarios with multiple data points (or an entire class), we propose an iterative divide-and-conquer algorithm. The datasets to be unlearned are clustered and a small number of representative data points are selected for unlearning. By iterating the above steps, the effect of unlearning can gradually cover the whole dataset to unlearn with only a few data points. This idea is also extended to the application of unlearning an entire class.
- We evaluated the effectiveness of our approach on multiple categorical datasets². The results show that both goals are achieved. Meanwhile, it is competitive in terms of efficiency and memory consumption in comparison with various baseline unlearning methods. We further show that the model obtained after executing our unlearning algorithm can successfully defend the membership inference attack, i.e., the unlearned data is considered not involved in the training process anymore.

Roadmap. In Section 2, we introduce machine unlearning and neural network repair, and formally link their goals. In Section 3, we consider a typical scenario of machine unlearning and introduce the corresponding threat model involving data holders, model owners, and auditors. In Section 4, we propose a novel unlearning approach PRUNE and explain its technical details. We evaluate the proposed approach via extensive experiments in Section 5. We provide further discussion in Section 6, list some of the challenges that PRUNE can address, and provide an outlook on future research. In Section 7, we review the work related to our approach. Finally, Section 8 concludes our work.

2. Preliminaries

2.1. Machine unlearning

As a line of research on AI privacy, machine unlearning has a variety of research objects, but the core issue is how to achieve the effect of data withdrawal. In this paper, we focus on how to unlearn data on Deep Neural Networks (DNNs) used for classification tasks. This is a very common task in many real-world scenarios. Formally, let $D = \{x_i, y_i\}_{i=1, \dots, n}$ be the dataset containing data points x_i and corresponding labels y_i . Given a DNN model $M : \mathcal{X} \rightarrow \mathcal{Y}$, \mathcal{X} is the input domain and $\mathcal{Y} = \{1, 2, \dots, L\}$ is the set of category labels. M_D is a DNN model trained on the dataset D . It makes judgment about the label of $x_i : \arg \max_{l \in \mathcal{Y}} M_D^l(x_i)$, where

² The code and data are released at (Li, 2024)

$M_D^l(x)$ denotes the output probability that model M_D considers the label of x to be l . When there are requests for erasure, $D_U = \{x_u, y_u\}_{u=1, \dots, r}$ denotes the set of data points to be unlearned and $D_U \subset D$. $D_R = D/D_U$ is the set of remaining data points. M_U is the model after the execution of the unlearning algorithm.

2.2. Neural network repair

Neural network repair is the line of research aimed at fixing different kinds of ‘errors’ of a neural network by modifying its parameters or architecture. Neural network repair is mainly applied in two scenarios. One is when a neural network is trained normally and the output accuracy is limited. The repair operation can improve the overall performance of the model. The other is when the neural network is maliciously attacked during training or while in use (Gu et al., 2017; Szegedy et al., 2014). The neural network cannot properly handle these disturbances, in which case repair is needed to correct the model’s output. Compared to traditional program repair, fault localization for black-box and unexplainable neural networks (Samek et al., 2017) is more challenging which makes repair efforts difficult in general.

Existing repair methods mainly include retraining/fine-tuning, modifying parameters (Sun et al., 2022; Usman et al., 2021) and patching network (Fu & Li, 2022; Sotoudeh & Thakur, 2019, 2021). Next, we formalize the neural network repair problem. Assume M behaves abnormally on the buggy set $X_R \subset \mathcal{X}$ whose correct output labels are stored correspondingly in $Y_R \subset \mathcal{Y}$. The goal of neural network repair is to obtain a repaired model M_R with two objectives in mind. First, M_R should be able to make correct prediction on the buggy set, which can be formalized as for any $x_r \in X_R$,

$$Obj_1^r : M_R(x_r) = y_r \quad (1)$$

The other objective is to make sure that the repaired model should maintain good performance on the remaining data:

$$Obj_2^r : \min \frac{1}{|D|} \sum_{i=1}^{|D|} |\arg \max_{l \in \mathcal{Y}} M_R^l(x_i) - \arg \max_{l \in \mathcal{Y}} M_D^l(x_i)| \quad (2)$$

where D is the data set that does not need to be repaired.

2.3. Linking unlearning and repairing

From the data holder’s perspective, unlearning is considered to be effective intuitively on x_u iff $\arg \max_{l \in \mathcal{Y}} M_U^l(x_u) \neq y_u$, i.e., the unlearned model can no longer make the correct judgement on his/her data. This objective can be formalized as for any $x_u \in D_U$,

$$Obj_1^u : \exists l \neq y_u, M_U^l(x_u) - M_U^{y_u}(x_u) > 0 \quad (3)$$

Note that Obj_1^u and Obj_1^r are two exactly mirrored operation.

Besides the unlearning objective in Eq. (3), similar to the repair task, we also have to pay attention to the overall performance of the model on the remaining data. If there is a significant decrease in the model performance after unlearning, the algorithm is of no practical value. The ideal situation is that the model produces no change in label judgments on D_R . Thus, the other objective in Eq. (4) is to make the performance change on the remaining data as small as possible. It is formulated as

$$Obj_2^u : \min \frac{1}{|D_R|} \sum_{i=1}^{|D_R|} |\arg \max_{l \in \mathcal{Y}} M_U^l(x_i) - \arg \max_{l \in \mathcal{Y}} M_D^l(x_i)| \quad (4)$$

When both objectives are achieved, the unlearning algorithm can be considered as meeting both the needs of data holders and the interests of the model owner. Our approach is designed precisely in accordance with these objectives. Beyond the apparent symmetry in objectives, framing machine unlearning as a neural network repair problem provides additional methodological insights. In particular, the repair perspective emphasizes localized and targeted model modification, rather than global

retraining, which aligns with the practical requirements of efficient unlearning. Neural network repair has extensively studied fault localization and minimal intervention principles, which naturally translate to identifying and modifying the parameters most responsible for memorizing the data to be removed. This connection not only unifies the objectives of unlearning and repair, but also motivates the design of unlearning algorithms that are controllable, efficient, and less disruptive to the model’s overall behavior.

3. Threat model

3.1. Certifiable unlearning

Definition 1 (Certifiable Unlearning). Given a trained model M_D and a data withdrawal request for a set of samples $D_U \subset D$, an unlearning mechanism produces an updated model M_U . We say that the unlearning mechanism is *certifiable* under a specified threat model if there exists a verification procedure \mathcal{V} such that a third-party auditor, with limited access to M_U and without access to the original training process or full model parameters, can verify with high confidence that:

1. the model’s predictive behavior on D_U has been sufficiently altered to satisfy the requested unlearning degree; and
2. the model’s predictive behavior on the remaining data $D \setminus D_U$ is largely preserved.

The verification is performed based solely on observable model behaviors (e.g., outputs to queries) and does not rely on retraining the model or accessing its internal parameters.

3.2. Participants and capabilities

We consider a typical scenario of machine unlearning to handle a data withdrawal request. In this scenario, three parties are involved: the data holders, the model owner, and a third-party auditor (e.g., government officials or certification providers). Their relationship is illustrated in Fig. 1. The model owner uses the data from the data holders for training the model. According to the GDPR (European Union, 2016) and other laws, the data holders are allowed to make data withdrawal requests. The model owner is supposed to process these requests within a certain time frame and avoid the privacy risks associated with data erasure. However, in the machine unlearning scenario, there exists a natural conflict of interests between the model owner and the data holders. That is, withdrawal of data can lead to potential degradation of model performance on the remaining data or increase in computational cost (or both). Subjectively, model owners tend to avoid such operations, which is why auditors are necessary. In addition, since some models are inaccessible to ordinary users, auditors can act as a third party to safeguard the legitimate interests of data holders. Specifically, the objectives, capabilities and background knowledge of the three parties are as follows. The component of risk assessment in Fig. 1 refers to the auditor’s evaluation of whether the unlearning process introduces potential privacy risks and corresponds to the privacy analysis based on membership inference conducted in our experiments.

Data holders. The goal of data holders is to have free reign over their data. That is, to decide whether their data are used by the model. Their ‘‘right to be forgotten’’ is mainly reflected in the request for data withdrawal at any point of time. As ordinary users, they often have difficulty accessing the specific parameters of the model.

Model owner. The goal of the model owner is to invoke the unlearning algorithm on the trained model to periodically process the data withdrawal requests. The overall performance of the model after performing the unlearning operation should not show a significant degradation and should provide clear evidence to the data holder or the auditor. Acts usually specify that there is a buffer time after the model owner receives a data withdrawal request. It is assumed that during this time, the model

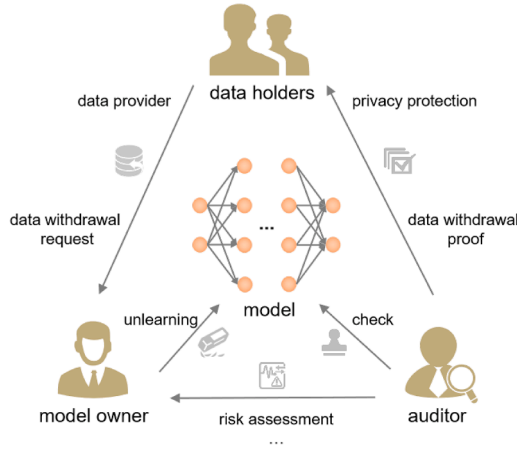


Fig. 1. There are three parties involved in the threat model: the data holders, the model owner, and the auditor.

owner still has access to the data to be erased. The case where the model owner outsources the training process is not considered here. That is, the model owner owns all the parameters of the model and can modify them directly.

Auditor. The goal of the auditors is to safeguard the legal rights of the data holders to the maximum extent. In a machine unlearning scenario, rights include not only the timely processing of data withdrawal requests, but also the avoidance of data leakage risks. The auditors might be allowed access to the model to a certain extent, but does not have complete knowledge of the model’s parameters. They have to judge whether unlearning is successful based on the data withdrawal request and the evidence provided by the model owner. In addition, they need to assess the privacy risks that may result from the operation of unlearning. As a third party, the auditors are responsible for both the data holders and the model owner to ensure that the data is legally applied.

Dishonest model owner assumption. We explicitly consider a threat model in which the model owner may behave dishonestly. In such cases, retraining-based unlearning strategies are insufficient for certification, as retrained models often exhibit outputs that are indistinguishable from the original model due to generalization effects (Thudi et al., 2022b). Consequently, auditors relying solely on output similarity to a retrained model may fail to detect whether unlearning has actually occurred. This limitation motivates the need for *certifiable unlearning* mechanisms whose effects can be externally verified through observable behavioral changes rather than internal parameter inspection. The detailed verification procedure is provided in Section 4.3.

4. Unlearning via repairing

In this section, we first introduce the properties that should be achieved for an ideal unlearning operation following our threat model. Then, we present our detailed patching-based repair framework PRUNE to achieve these properties considering multiple practical settings: 1) unlearning a single data point; 2) unlearning a set of data points; and 3) unlearning an entire category. Lastly, we compare PRUNE with related unlearning techniques qualitatively.

4.1. Unlearning properties

A qualified machine unlearning mechanism should meet the following properties.

1) *Utility on D_U .* This is the most important property from the perspective of data holders and the auditors. When the model loses its prediction ability on the data point to be withdrawn, the data point is considered to be successfully erased. It is an intuitive and easy-to-test metric

for the auditor who does not have access to the full parameters of the model.

2) *Efficiency.* Achieving data withdrawal is a legal requirement for the model owner. However, computational costs should be taken into account. The computational cost here includes the normal training phase and the unlearning phase. When the computational cost of an unlearning algorithm is much higher than that of retraining, the model owner may also make the decision to abandon the model. An unlearning algorithm with realistic applicability should be competitive with retraining in terms of efficiency.

3) *Overall performance.* This is the attribute that model owners care about most. Neural network models are widely used because of their excellent performance. It is unacceptable if the performance of the model deteriorates by performing unlearning operation. Therefore, for multi-point unlearning, we require that the unlearned model still guarantees high accuracy on the test set, meaning the model has not lost its generalization ability. For the case of class unlearning, we require that the model performs well on test set in addition to the target category data.

4) *Privacy.* The “right to be forgotten” is a right that is generated around the privacy of data holders. A reasonable machine unlearning mechanism should protect the privacy of the user even after the algorithm is executed. For example, privacy attack algorithms such as membership inference attacks should not be able to tell that the data has been involved in the training process based on the unlearning model. This property, as an implicit property, should be jointly safeguarded by the model owner and the auditor.

5) *Fairness of rights.* Each data holder should have equal rights on their data. In practice, a data holder may make a withdrawal request independently or jointly with all data holders belonging to the same category. These two cases correspond to multi-point and class unlearning, respectively. Data withdrawal requests are randomized on the training set. The distribution of data should not have an impact on the actual withdrawal effect when the mechanism is designed. Emphasizing the sequential selectivity of the unlearning set can make the mechanism fail in some cases.

4.2. The PRUNE framework

In the following, we present our Patching based Repair framework for certifiable machine UNlearning (PRUNE) in details. As shown in Fig. 2, the key idea of PRUNE is to generate a targeted “patch” network on the original model M_D training on D to unlearn the specified data, i.e., redirecting the model’s prediction to elsewhere wrong. The patch is *targeted* in the sense that there is a one-to-one mapping from the specified data point to unlearn x_u to the patch network $c(x)$. And the patch network will only be activated when running the model on the specific data point, which means the model’s performance will not be affected on any other data than the data to unlearn. In the following, we present the technical details of how we realize the idea of PRUNE and generate the patch network for different unlearning scenarios.

We denote a DNN model by the concatenation of two sequential parts $M = M_p \oplus M_c$, where M_p is used for feature extraction with operations like convolution and M_c are the fully connected layers. In general, our approach is applicable to continuous piecewise linear (CPWL) neural networks, i.e., M_c with Rectified Linear Unit (ReLU) activation function $\text{ReLU}(x) = \max(0, x)$. We do not have requirements on M_p . The proof details of the theorems are provided in Li (2024).

Theorem 1. Given a neural network model $M_D = M_p \oplus M_c$ trained on D , and a sample data x_u to unlearn, it is guaranteed that we can construct a patch network c_S for M_c and obtain an unlearned model $M_U = M_p \oplus (M_c + c_S)$ such that: 1) $M_U(x_u) \neq M_D(x_u)$; and 2) for $x \in D/x_u$, $M_U(x) = M_D(x)$.

Next, we introduce how the patch network c_S is constructed. In general, a *patch network* consists of two parts: a *confusion sub-network*

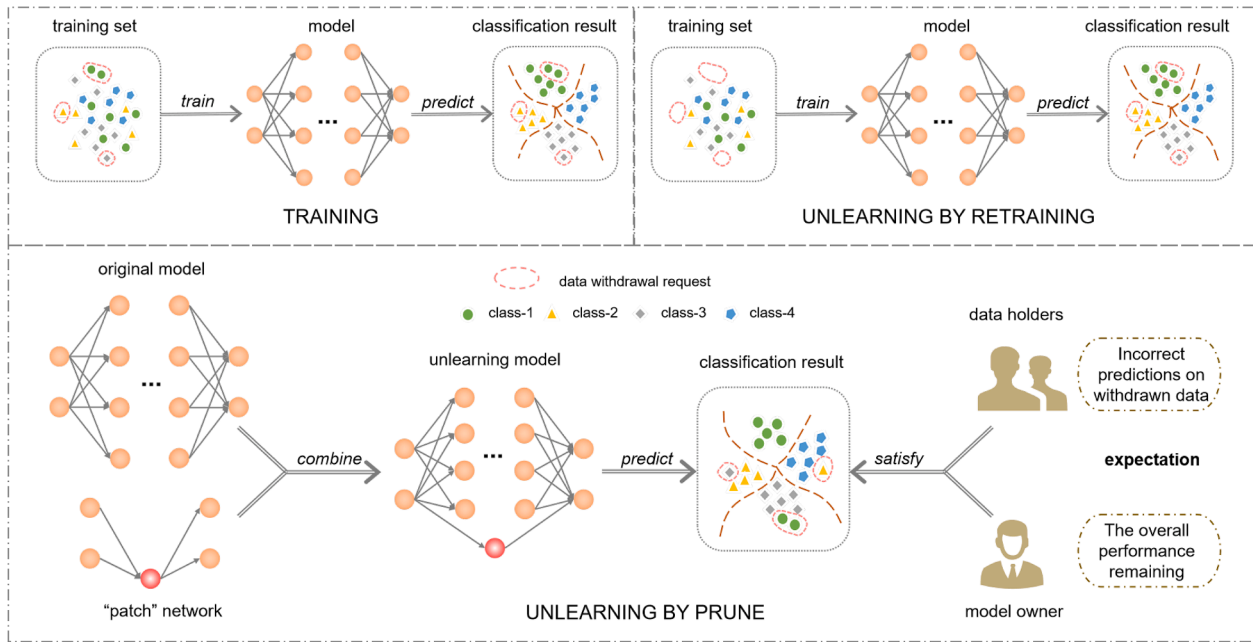


Fig. 2. The framework of PRUNE. When data withdrawal requests occur, PRUNE reduces the model's ability to judge these specific data points by generating a patch network. This mechanism solves the problem that the generalization ability of the model makes the unlearning process impossible to verify intuitively. At the same time it avoids the degradation of the overall performance of the model. PRUNE meets the needs of data holders and model owners.

that affects the output domain (directing the prediction on x elsewhere) and a *support sub-network* that limits the side effect (not affecting the model's prediction on the remaining data). The construction has three steps whose details are as follows.

1) *Locating the linear region of x_u .* Considering our unlearning goal in Eq. (3), we should pay more attention to the correspondence between the input domain and the output domain, while ignoring the model structure change. Since our study object is CPWL, the linear region where the data point x_u to be unlearned lies can be first computed similarly to Lee et al. (2019).

Lemma 1. Given a CPWL neural network with neurons z , each $z_j^i \in z$ induces a feasible set $S_j^i(x)$ for input $x_u \in \mathcal{X}$. For $\bar{x}_u \in \mathcal{X}$,

$$S_j^i(x_u) = \begin{cases} \left(\begin{matrix} \left(\nabla_{x_u} z_j^i \right)^T \bar{x}_u + z_j^i - \left(\nabla_{x_u} z_j^i \right)^T x_u \geq 0, z_j^i \geq 0 \\ \left(\nabla_{x_u} z_j^i \right)^T \bar{x}_u + z_j^i - \left(\nabla_{x_u} z_j^i \right)^T x_u \leq 0, z_j^i < 0 \end{matrix} \right) & (5) \end{cases}$$

z_j^i denotes the i th neuron (before activation) in the j th hidden layer and $\nabla_{x_u} z_j^i$ is the sub-gradient calculated by back-propagation. The linear region including x_u is the feasible set $S(x_u) = \cap_{i,j} S_j^i(x_u)$.

Assume that $S(x_u) = \{a_i x_u \leq b_i\}_{i=1,2,\dots,N}$ is a linear region calculated by Lemma 1, which is composed of N inequalities. To achieve the goal in Eq. (4) (not affecting the remaining data), we further use a support network to restrict the subsequent unlearning effect to occur only on $S(x_u)$ defined by the following lemma (Fu & Li, 2022).

Lemma 2. For a neural network using ReLU function as the activation function, the support network on the given feasible set $S(x_u)$ is defined as

$$n_S(x_u, \lambda) = \text{ReLU} \left(\sum_i n(b_i - a_i x_u, \lambda) - N + 1 \right) \quad (6)$$

where $n(x_u, \lambda) = \text{ReLU}(\lambda x_u + 1) - \text{ReLU}(\lambda x_u)$ and λ is a parameter to control the boundary of $n(x_u)$.

A more intuitive illustration of the support network activation is provided in Appendix A.

2) *Optimizing the confusion network.* Obj_1^u requires M_U to fail to predict on x_u . This is the most basic utility of the confusion network $m(x_u)$,

i.e., $M_U(x_u) = M_p(x_u) \oplus (M_c(x_u) + m(x_u)) \rightarrow \hat{y}_u \neq y_u$. Meanwhile, $m(x_u)$ is expected to be minimal in the function space to reduce the impact on the model performance. Thus $m(x)$ for unlearning x_u can be optimized based on the following Lemma.

Lemma 3. Given a neural network model $M_D = M_p \oplus M_c$, and a sample data $x_u \in \mathcal{D}$ with the feasible set $S(x_u)$, the optimization objectives of the confusion network $m(x) = Cx + d$ can be formalized as

$$\begin{cases} \min_{C,d} \max_{x \in S(x_u)} |m(x)| \\ M_U(x) = M_p(x) \oplus (M_c(x) + m(x)) \\ \forall x \in S(x_u), M_U^{\hat{y}_u}(x) - M_U^l(x) > 0, l \neq \hat{y}_u \in \mathcal{Y} \end{cases} \quad (7)$$

where C is a matrix, d is a vector, and \hat{y}_u is the confusing label randomly taken in $\mathcal{Y} \setminus y_u$.

Eq. (7) can be converted to a linear programming (LP) problem by enumerating the vertices and be solved by robust optimization (Ben-Tal et al., 2009).

3) *Combining into patch network.* For a single data point x_u in S to unlearn, the patch network $c(x_u)$ can be expressed as

$$c_S(x_u) = \text{ReLU}(m_S(x_u) + H \cdot n_S(x_u, \lambda) - H) - \text{ReLU}(-m_S(x_u) + H \cdot n_S(x_u, \lambda) - H) \quad (8)$$

where H is the upper bound of $m(x)$ obtained in the linear region S .

Based on the above steps, Theorem 1 provides ideal and certifiable unlearning guarantee for a given data point by producing a targeted patch. This is highly desirable to precisely remove a small amount of data points. While the theorem is stated for CPWL networks, we provide formal extensions to arbitrary backbones and smooth activations in D. Next, we further illustrate how to efficiently apply PRUNE to handle unlearning of multiple data points and entire class cases respectively.

Unlearning on multiple data points

Note that the main cost of our algorithm is on the optimization of Eq. (7). Considering the time complexity, for the dataset \mathcal{D}_U to unlearn, we expect to optimize fewer confusion networks to affect more labels of data points. $m(x)$ is optimized based on a linear region where a single x_u is located, so our intuition is to choose the most representative data

points to generate $m(x)$. Here, we use clustering to select representative data points as follows. The data points $\{x_u\}_{u=1,\dots,r}$ are clustered to obtain K centroids $\{x_c^k\}_{k=1,\dots,K}$. The optimized objective formula is

$$\arg \min_{D_U} \sum_{k=1}^K \sum_{x_u^k \in D_U} \|x_u^k - x_c^k\|^2 \quad (9)$$

Since x_c^k has the shortest Euclidean distance from all points in their cluster $\{x_u^k\}_{u=1,\dots,r}$, we use x_c^k as the representative point to generate confusion network m_k according to Eq. (7). We feed $\{x_u^k\}_{u=1,\dots,r}$ into the temporary network $M_D + m_k$ to test whether the label of x_u^k can still be judged correctly. If the label has been misjudged, the corresponding support network n_u^k is calculated according to Eq. (6). If the output of this point has not changed, it will be recorded into the remaining unlearning dataset D_{UR} for a new round of iteration. After traversing all points in D_U^k , a confusion network m_k and a series of support networks $\{n_u^k\}_{u=1,\dots,r}$ are obtained, so the corresponding patch network c_k can be calculated by

$$c_k(x, \lambda) = \text{ReLU} \left(m_k(x) + \max_{1 \leq u \leq r} \{n_u^k(x, \lambda)\} \cdot H_u - H_u \right) - \text{ReLU} \left(-m_k(x) + \max_{1 \leq u \leq r} \{n_u^k(x, \lambda)\} \cdot H_u - H_u \right) \quad (10)$$

where $H_u = \max \{|m_k(x)| | x \in \cup_{1 \leq u \leq r} S(x_u^k)\}$. And when all clusters in D_U have been optimized m_k , tested, generated $\{n_u^k\}_{u=1,\dots,r}$ and calculated c_k , M_c can be updated to $M_c + \{c_k\}_{k=1 \rightarrow K}$. Finally, we judge whether to end the entire iterative process by unlearning success rate $1 - D_{UR}/D_U$. When it is higher than the required unlearning degree δ , the algorithm is completed. The overall process of PRUNE for multipoint unlearning is summarized in Algorithm 1 which is guaranteed to terminate and we thus have the following theorem. The proof of Theorem 2 is provided in C.

Theorem 2. Given a neural network model $M_D = M_p \oplus M_c$ trained on dataset D , a target unlearning set $D_U \subset D$, and an unlearning degree $\delta \in (0, 1)$, PRUNE constructs a finite set of patch networks $\{c_k\}_{k=1}^K$ and produces an updated model $M_U = M_p \oplus (M_c + \sum_{k=1}^K c_k)$ such that the following unlearning condition holds:

$$\frac{|\{x \in D_U : M_U(x) \neq M_D(x)\}|}{|D_U|} \geq \delta.$$

Moreover, the construction is guaranteed to terminate after a finite number of iterations.

Unlearning on an entire class

For classification tasks, another common scenario is to remove an entire class of data. When the data holders with the same label in the dataset jointly make a data withdrawal request for some reason, the process of PRUNE on the model should change. This is because the model should not be able to predict the data in a category well if the data in that category have not been involved in the training process of the model. Next, we illustrate the application of PRUNE in this setting. Suppose $D_U = \{x_u, y_{\text{unlearn}}\}_{u=1,\dots,r}$ and y_{unlearn} is the label of the category to be unlearned. First, the most representative points are selected without clustering because the data in the same category have similarity in feature distribution. After finding the centroid x_c in this category directly by Euclidean distance, m_c corresponding to x_c is optimized according to Eq. (7). Afterwards, similar to multi-point unlearning, the output of x_u on $M_p \oplus (M_c + c)$ is compared to determine if the erasure effect $y_{\text{unlearn}} \rightarrow \hat{y}_{\text{unlearn}}$ is covering x_u . However, in limiting the side effects of PRUNE, the entire class of unlearning has additional requirements, so the computation of support networks are different from random multipoint unlearning. That is, we want m_c to take effect not only for D_U , but also generalize for all data points labeled with y_{unlearn} . To achieve better generalization effect, we further add perturbation Δ on x_c . With the help

Algorithm 1 PRUNE-multipoint.

```

1: Input:  $D_U, M_D = M_p \oplus M_c$ 
2: Output:  $M_U$ 
3: Initialize clusters number  $K$ 
4:  $D_U^k, \{x_c^k, y_c^k\}_{k=1 \rightarrow K} \leftarrow \text{KMeans}(D_U, K)$ 
5: for  $k \leftarrow 1$  to  $K$  do
6:    $\hat{y}_c^k \leftarrow \text{Randomize } \mathcal{Y} \setminus y_c^k$ 
7:    $m_k \leftarrow$  according to Eq. (7)
8:   for  $(x_u^k, y_u^k) \in D_U^k$  do
9:     if  $M_p(x_u^k) \oplus (M_c(x_u^k) + m_k(x_u^k)) \neq y_u^k$  then
10:       $n_u^k \leftarrow$  according to Eq. (6)
11:     else
12:        $D_{UR} \leftarrow (x_u^k, y_u^k)$ 
13:     end if
14:   end for
15:    $c_k \leftarrow$  according to Eq. (10)
16: end for
17:  $M_c \leftarrow M_c + \{c_k\}_{k=1 \rightarrow K}$ 
18: repeat
19:    $D_U \leftarrow D_{UR}$ 
20:   line4-line17
21: until  $1 - D_{UR}/D_U > \delta$ 
22:  $M_U = M_D$ 

```

of CROWN (Zhang et al., 2018), a verification tool for bound propagation, the perturbation of upper boundary $\bar{\Delta}$ and lower boundary $\underline{\Delta}$ that can satisfy $M_D(x_c + \Delta) = y_{\text{unlearn}}$ is quickly computed. We use Eq. (5) to calculate the activation pattern $S(x_c)$ on M_D by bringing in $x_c + \bar{\Delta}$ and $x_c + \underline{\Delta}$ to obtain a more relaxed activation pattern $S'(x_c + \Delta)$. Based on $S'(x_c + \Delta)$, the corresponding n'_c can be generated according to Eq. (6). The same process is used to generate n'_u for the other data points in D_U . Finally for the entire class of patch network $c(x, \lambda)$ is calculated by

$$c(x, \lambda) = \text{ReLU} \left(m_c(x) + \max_{1 \leq u \leq r} \{n'_u(x, \lambda)\} \cdot H_u - H_u \right) - \text{ReLU} \left(-m_c(x) + \max_{1 \leq u \leq r} \{n'_u(x, \lambda)\} \cdot H_u - H_u \right) \quad (11)$$

where H_u is calculated in the same way as Eq. (8). The overall process of PRUNE for class unlearning is summarized in Algorithm 2. A more detailed discussion of the termination criterion and convergence behavior is provided in Appendix C.

Having introduced the PRUNE framework and its theoretical guarantees, we next describe how the resulting unlearning behavior can be externally verified by a third-party auditor under realistic access constraints.

4.3. Verification procedure for certifiable unlearning

Based on the threat model in Section 3, we consider a third-party auditor with black-box access to the unlearned model M_U . Given a data withdrawal request for a set D_U and a desired unlearning degree δ , the auditor verifies whether unlearning has been correctly executed through the following procedure.

- **Query with withdrawn data.** For each data point $x \in D_U$, the auditor queries the deployed model M_U multiple times and records the predicted outputs.
- **Behavioral change test.** The auditor estimates the empirical probability that the prediction of M_U on x differs from the reference behavior (e.g., the original model prediction or the ground-truth label), and checks whether the empirical ratio exceeds the specified threshold δ .
- **Preservation test on remaining data.** The auditor samples a subset of data points from $D \setminus D_U$ and verifies that the predictive behavior

Algorithm 2 PRUNE-class.

```

1: Input:  $D_U, M_D = M_p \oplus M_c, y_{unlearn}$ 
2: Output:  $M_U$ 
3:  $\hat{y}_{unlearn} \leftarrow \text{Randomize } \mathcal{Y} \setminus y_{unlearn}$ 
4:  $x_c \leftarrow \text{centroid } D_U$ 
5:  $m_c \leftarrow \text{according to Eq. (7)}$ 
6: for  $x_u \in D_U$  do
7:   if  $M_p(x_u) \oplus (M_c(x_u) + m_c(x_u))! = y_{unlearn}$  then
8:      $\bar{\Delta}, \underline{\Delta} \leftarrow \text{CROWN}(M_D, y_{unlearn}, x_u)$ 
9:      $S'(x_u + \Delta) \leftarrow \text{according to Lemma 1}$ 
10:     $n'_u \leftarrow \text{according to Eq. (6)}$ 
11:   else
12:      $D_{UR} \leftarrow x_u$ 
13:   end if
14: end for
15:  $c \leftarrow \text{according to Eq. (11)}$ 
16:  $M_c \leftarrow M_c + c$ 
17: repeat
18:    $D_U \leftarrow D_{UR}$ 
19:   line3-line16
20: until  $1 - D_{UR}/D_U > \delta$ 
21:  $M_U = M_D$ 

```

of M_U on these samples remains consistent with the original model, up to a small tolerance.

If both the behavioral change test on D_U and the preservation test on $D \setminus D_U$ are satisfied, the auditor accepts that the unlearning request has been correctly executed.

4.4. Comparison with related techniques

Table 1 compares PRUNE with existing unlearning methods. The comparison mainly revolves around the following dimensions: (1) Use of Remaining Data. Whether the unlearning algorithm will use the remaining training data again. (2) Use of Erasure Data. Whether to apply the unlearning algorithm on the data to withdraw. (3) Single Point. Whether the unlearning algorithm can precisely forget a single data point. (4) Model Training Recording. Whether the unlearning algorithm needs to record relevant information during machine learning model training, such as gradients. (5) Certifiability. Whether the unlearning algorithm has theoretical guarantees. And the data holders or auditor can verify the erasure effect of the data requested to be withdrawn. (6) Limited Side Effect. After performing the unlearning algorithm, whether the performance of the model on the remaining data is retained. The last two dimensions are quantitatively evaluated in Section 5.2, where certifiability is measured by ΔA_u , and the side effects are measured by ΔA_{res} and ΔA_{test} .

PRUNE erases data by adding certifiable minimal “patches” to the original neural network. No additional access to the remaining training dataset is required. PRUNE provides intuitively measurable forgetting effects while keeping the model’s performance on the remaining data barely changed. Unlike AML, PRUNE provides deterministic theoretical support in addition to the easily measurable effects. Compared with unlearning methods that need to record model training information, PRUNE only performs lightweight post-processing on a given neural network model.

5. Experiments

In this section, we describe the experimental setup and conduct extensive experiments aiming to answer the following research questions:

- **RQ1:** Can our approach forget specific data points while keeping the model performance as constant as possible?

Table 1

Comparison with existing unlearning techniques.

	PRUNE	Retrain	SISA	AML	FU
Use of Remaining Data	✗	✓	✓	✓	✗
Use of Erasure Data	✓	✗	✗	✓	✓
Single Point	✓	✓	✓	✓	✓
Training Recording	✗	✗	✓	✓	✗
Certifiability	✓	✗	✗	✓(effect)	✓
Limited Side Effect	✓	✓	✗	✗	✗

Table 2

Details of datasets and models.

Dataset	Classes	Features	Training Set	Test Set	Model
Purchase-20	20	600	38,758	9689	FC(256)
HAR	6	561	8238	2060	FC(256)
MNIST	10	28 × 28	60,000	10,000	FC(256 × 256)
CIFAR-10	10	32 × 32 × 3	50,000	10,000	VGG16

- **RQ2:** How does the efficiency of our approach compare with baseline methods?
- **RQ3:** Can our approach affect membership inference against erased data?
- **RQ4:** How well does our approach perform with unlearning on the entire class of data?
- **RQ5:** How our approach is affected by hyperparameters?

5.1. Experimental setup

Datasets and Models. We conduct experiments on four popular classification datasets in machine unlearning research: Purchase-100 (Shokri et al., 2017), Human Activity Recognition (HAR) (Bulbul et al., 2018), MNIST (LeCun et al., 1998), and CIFAR-10 (Krizhevsky et al., 2009). The fully connected (FC) neural network with ReLU activation function is chosen to perform classification prediction on the first three datasets. Details of the dataset and models are in Table 2. Considering the application scenario and Property 5 requirement of unlearning, the data points in D_U are randomly selected from the training set. The parameters for models training are described in Li (2024).

Baselines. We compare our approach with four widely used unlearning mechanisms: 1) Retraining, as the naive unlearning method, trains the model from scratch on $D \setminus D_U$. 2) SISA (Bourtole et al., 2021) shards the training data and then trains them separately. It yields predictions obtained by submodel majority voting. Each shard is further sliced, and the model checkpoint is stored during training for each slice, allowing for the retraining of a new model from an intermediate state. 3) Amnesiac Machine Learning (AML) (Graves et al., 2021) records the gradient data of each batch during the training phase. If unlearning is to be performed, the gradients of the affected batches are directly removed to save the cost of retraining. However, in order to recover the model performance to some extent, AML has to perform easy training again after removing the gradient. 4) Features Unlearning (Warnecke et al., 2023) performs closed updates of model parameters based on influence functions to forget specific training data labels or features. Considering the negative impact on model performance, we use the second-order update method for FU.

5.2. Utility guarantee

We evaluate each unlearning method using three metrics that reflect both effectiveness and side effects: 1) the change in accuracy on the test set ΔA_{test} , 2) the change in accuracy on the retained data not requested for unlearning ΔA_{res} , and 3) the accuracy drop on the unlearned data, computed as $\Delta A_u = A_{u_b} - A_{u_a}$, reflects the extent of forgetting, where

A_{u_b} and A_{u_a} represent the model’s accuracy on the unlearned data before and after unlearning, respectively. A larger ΔA_{u_i} indicates a more effective forgetting process, addressing the data holder’s concern. Meanwhile, smaller ΔA_{tes} and ΔA_{res} suggest lower negative impact on the model’s generalization and retained knowledge.

When applying PRUNE for single-point unlearning, we can achieve complete forgetting effects as promised by [Theorem 1](#) (Tables are omitted being boring). For unlearning multiple data points, [Table 3](#) shows the results of the model accuracy on different datasets using different unlearning algorithms. The number of data points in D_U varies between [100, 200, 300, 500]. Data points in D_U are sampled randomly from different training batches. The effectiveness evaluation for PRUNE are twofold: 1) the model loses its prediction accuracy for the data to be erased and 2) the overall performance of the model is not compromised.

In terms of erasure effectiveness, we observe that retraining and SISA lead to almost no change in the model’s predictions on the unlearned dataset D_U , even after the unlearning procedure is executed. FU performs slightly better, but the reduction in prediction accuracy remains limited. This phenomenon can be attributed to the generalization capability of trained models: even if a specific data point is removed from the training set, the model may still produce similar outputs due to its ability to infer patterns from similar examples. However, from the perspective of data holders and third-party auditors, such limited behavioral changes provide no intuitive or reliable indication that unlearning has actually been conducted. In contrast, both AML and PRUNE exhibit a significant performance drop on D_U , with a large ΔA_{u_i} exceeding 90%. This substantial change clearly demonstrates their ability to disrupt the model’s learned behavior on the target data, effectively erasing its memory of the withdrawn samples. Importantly, this shift can be easily verified from the outside: an auditor can simply query the model with previously submitted data and observe the output discrepancy. If the model no longer returns correct predictions for these inputs, this serves as a strong, intuitive signal that forgetting has occurred. In this sense, our approach aligns well with real-world needs-data holders can obtain a tangible and individualized guarantee of unlearning by observing model failure on their own data points. It is also worth noting that in PRUNE, the final A_{u_a} does not reach zero. This is not due to a limitation in the algorithm’s capability but rather a result of a predefined termination condition used to balance erasure effectiveness and computational cost. In practice, this stopping criterion can be adjusted to trade off between unlearning strength and resource efficiency as needed.

In terms of overall performance, we observe that AML exhibits a large ΔA_{tes} as the number of forgotten data points increases, indicating a notable degradation in test accuracy despite effective unlearning. For example, AML suffers from 12.59% drops on the accuracy of the test set for MNIST after unlearning 500 samples. It is due to the fact that AML directly removes the gradient information from the data points, which can cause catastrophic forgetting of the model. Even with recovery training, it is difficult to regain initial performance. For FU, unlearning also results in a sharp decrease in accuracy on the testing set and the unwithdrawn training set. This is because while FU uses influences functions to control the model parameter updates, it unavoidably affects the model’s prediction for other data when forgetting specific data points. The other two baseline methods, on the other hand, performed normally on A_{tes} . The lack of a small fraction of data does not lead to a large impairment in the performance of the model when trained from scratch. The performance degradation of SISA on the HAR dataset is due to the over-representation of the withdrawn data in the training set of the sub-model. PRUNE minimizes the degradation of the overall performance of the model by making $m(x)$ effective only on D_U . It maintains a near-zero ΔA_{tes} on the Purchase, HAR, and MNIST datasets, while the degradation on CIFAR-10 remains within 4%. This indicates that our approach is effective in terms of overall performance maintenance. Additionally, the accuracy of the model obtained by PRUNE does not change significantly on the remaining data, further showing that the other data points involved in the training process are affected in a very limited way.

Remark 1. PRUNE achieves easily measurable and assured-level unlearning while not affecting the model performance much.

5.3. Efficiency comparison

In this section, we evaluate the efficiency of different unlearning methods. Since some unlearning algorithms involve additional operations during the training phase, we record the time required for both the training and unlearning phases of the model and then combine them. For a fair comparison, we normalized the execution time of all methods by the time to train from scratch (i.e., Retrain). The experimental results are shown in [Fig. 3](#). For the training efficiency, PRUNE and FU, which involve no additional computations, exhibit nearly identical performance to Retrain. In contrast, AML and SISA take significantly more time for training. Especially for AML, there is a positive correlation between the time spent and the quantity of unlearning data. This is attributed to the need for computing the parameter updates for each batch that includes the unlearning data during training. In practical scenarios, when the unlearning samples are completely random, it becomes essential to record all batch updates, resulting in a significantly increased time overhead. Furthermore, there are other limitations, such as its inapplicability to models that were not originally designed with data removal in mind, and the recording of training parameters may introduce new risks of information leakage. For the unlearning efficiency, FU has the best performance among all candidate methods due to its closed updating of the model. Excluding FU, PRUNE exhibits a clear efficiency advantage over the other three baselines on high-dimensional datasets. On MNIST, for instance, it requires only 83% of the standard training time to forget 300 data points. PRUNE possesses a notable advantage in terms of efficiency when applied to the VGG16 model, as it performs unlearning by selectively modifying only the fully connected layers. Combined with the training efficiency, PRUNE demonstrates its substantial potential in handling complex datasets and models.

Remark 2. PRUNE is competitive in terms of efficiency and is flexible as a post-processing method.

5.4. Privacy guarantee

We use the membership inference attack proposed in [Yeom et al. \(2018\)](#) to evaluate whether unlearning mechanisms can eliminate the contribution of specific data points in the training process. Recall is used as a success metric, which measures the proportion of data points to be unlearned that are identified to have participated in the training. This metric is indicative of the extent of information leakage after unlearning. The lower the recall rate, the better the unlearning effect of the mechanism. Details of the membership inference evaluation are provided in [Appendix B](#). From the results in [Table 4](#), we can observe that our approach is able to successfully deceive membership inference attacks. Moreover, combined with the effectiveness experiment, we find that the success of the membership inference attack is limited to data points not affected by our unlearning process. In contrast, retraining, SISA and FU do not substantially reduce the success rate of membership inference attack due to the overlapping distribution of data points in the dataset. This echoes the dilemma faced by data holders: the presence of overlapping distributed data points brings inherent difficulty in intuitively verifying that the unlearning algorithm is performed. Even the auditor as a third party cannot provide credible evidence to

Table 3
Performance comparison when different numbers of data points are unlearned.

Dataset	D_U	Acc(%)	Retrain	SISA	AML	FU	PRUNE	Dataset	D_U	Acc(%)	Retrain	SISA	AML	FU	PRUNE
Purchase-20	0	A_{tes}	95.08	90.32	95.16	92.76	95.30	HAR	0	A_{tes}	95.05	95.29	94.98	97.78	95.82
		A_{res}	98.37	91.39	99.44	99.56	98.38			A_{res}	96.34	95.89	97.74	98.79	97.57
	100	ΔA_{tes}	-0.09	0.05	3.51	12.07	0.00		ΔA_{tes}	-0.04	1.42	14.26	0.68	0.00	
		ΔA_{res}	-0.12	-0.02	2.02	17.99	0.25		ΔA_{res}	-0.07	1.26	0.77	0.33	1.10	
		ΔA_u	0.00	3.00	95.00	16.00	97.17		ΔA_u	0.00	1.00	100.00	0.00	92.67	
	200	ΔA_{tes}	0.11	-0.36	3.94	11.29	0.00		ΔA_{tes}	0.13	2.02	14.75	24.88	0.21	
		ΔA_{res}	-0.08	-0.12	2.56	17.60	0.49		ΔA_{res}	-0.02	1.85	1.36	24.27	1.34	
		ΔA_u	4.00	1.50	100.00	14.50	97.70		ΔA_u	1.00	2.50	99.00	18.50	91.17	
	300	ΔA_{tes}	-0.09	-0.35	4.17	13.52	0.00		ΔA_{tes}	0.15	4.27	14.46	16.84	0.32	
		ΔA_{res}	-0.10	-0.34	0.96	19.40	0.74		ΔA_{res}	0.19	3.84	1.01	17.56	2.34	
		ΔA_u	1.67	1.66	99.33	8.34	95.00		ΔA_u	1.33	5.34	97.33	16.33	92.45	
	500	ΔA_{tes}	0.01	0.22	4.66	10.02	0.00		ΔA_{tes}	0.03	7.30	14.22	23.79	0.71	
ΔA_{res}		-0.08	0.07	3.14	16.93	1.23	ΔA_{res}	0.16	8.12	0.72	25.25	3.50			
ΔA_u		3.60	1.20	100.00	10.00	95.49	ΔA_u	0.20	7.00	95.20	27.00	93.60			
MNIST	0	A_{tes}	98.04	96.05	96.37	98.43	98.00	CIFAR-10	0	A_{tes}	87.64	86.66	86.73	86.36	87.63
		A_{res}	99.89	96.14	99.11	99.90	99.64			A_{res}	97.81	97.29	96.03	99.45	97.46
	100	ΔA_{tes}	-0.13	0.04	9.53	-0.15	0.00		ΔA_{tes}	-0.49	0.05	11.59	2.64	1.04	
		ΔA_{res}	0.00	-0.01	1.16	-0.04	0.16		ΔA_{res}	-0.13	0.13	5.71	13.37	1.45	
		ΔA_u	1.00	1.00	95.00	0.00	96.60		ΔA_u	10.00	0.00	95.00	8.74	92.20	
	200	ΔA_{tes}	-0.10	-0.05	9.85	5.07	0.00		ΔA_{tes}	0.72	0.01	13.16	2.85	2.30	
		ΔA_{res}	0.00	-0.02	0.74	6.99	0.32		ΔA_{res}	1.10	0.12	8.60	13.95	2.81	
		ΔA_u	2.50	0.50	94.00	8.00	95.57		ΔA_u	7.50	1.30	93.38	9.69	90.30	
	300	ΔA_{tes}	-0.19	-0.05	12.22	7.13	0.00		ΔA_{tes}	0.48	-0.18	11.09	2.13	3.01	
		ΔA_{res}	0.00	-0.01	0.20	8.30	0.49		ΔA_{res}	1.05	-0.07	6.66	14.11	3.94	
		ΔA_u	1.33	0.33	97.67	7.34	98.00		ΔA_u	9.67	0.22	95.50	12.59	89.83	
	500	ΔA_{tes}	-0.02	0.07	12.59	0.72	0.00		ΔA_{tes}	0.13	0.08	12.11	5.24	3.98	
ΔA_{res}		0.00	-0.01	2.12	2.44	0.80	ΔA_{res}	0.40	-0.05	5.52	16.93	5.63			
ΔA_u		1.60	0.40	97.00	2.20	96.43	ΔA_u	11.20	0.80	96.85	14.00	93.20			

Table 4

The success rate (%) of member inference attack on D_U before and after using different unlearning mechanism when $D_U = 100$. The value indicates the proportion of D_U that the data is considered to be involved in the training process. Lower values show better unlearning effects.

Unlearning Mechanism	Purchase-20		HAR		MNIST		Cifar-10	
	before	after	before	after	before	after	before	after
Retrain	84.30	84.30	84.00	83.70	92.60	83.00	84.00	78.00
SISA	84.20	80.00	83.40	79.80	92.60	91.00	90.80	87.40
AML	99.00	0.00	89.70	0.00	77.50	0.00	73.30	0.00
FU	85.30	70.50	84.80	84.70	90.80	88.90	91.00	90.20
PRUNE	81.70	0.00	81.40	4.04	92.90	1.01	84.80	5.00

the data holders on this basis. In this regard, our method provides a more observable and verifiable forgetting effect, even under black-box settings.

5.5. Class processing

For unlearning data of the whole category, we consider the performance of the model both on D_U and on that category data in the test set. Table 5 shows more detailed statistics obtained by randomly selecting the category labels for unlearning 20 times on the four datasets. A_u and A_r denote the prediction accuracy of the model on the data of the category to be withdrawn and on the data of the rest categories in the training set. They are used to evaluate how well PRUNE implements the data withdrawal request and the side effects caused by the unlearning. $A_{tes,u}$ and $A_{tes,r}$ are the prediction accuracy of the model on the data in the same category as the data to be withdrawn in the test set and on the remaining categories in the test set. They are used to assess whether PRUNE can generalize the effect of class unlearning and its impact on the overall performance of the model. By comparing the changes in the four metrics before and after the execution of our algorithm, we can draw the basic conclusion that PRUNE is capable of the class unlearn-

ing task. With the zeroing of A_u , there is a sharp drop in $A_{tes,u}$ (over 80%), which implies that the model loses the ability to predict the target category. That is, PRUNE successfully achieves the goal of class unlearning. On the other hand, the generalization effect of PRUNE on class unlearning, $A_{tes,u}$, fluctuates depending on the dataset, e.g., the generalization effect on two structured datasets is better than that on two image datasets, which suggests that PRUNE should be set with more reasonable parameters in conjunction with the data in practice. Additionally, the model's prediction accuracy for the remaining categories across the four datasets is not much affected, with no more than a 5% decrease on $A_{tes,r}$. It indicates that PRUNE enabled the model to maintain the overall performance even if a certain category was unlearned.

5.6. Hyper-parameter Tuning

Varying the number of centroids K

Before executing our algorithm, it is necessary to determine the number of centroids K . That is, the number of representative data points chosen by clustering. Different K will lead to different centroids, which directly affects the subsequent patch network generation and the accuracy of the model on D_U . Furthermore, K also determines the number

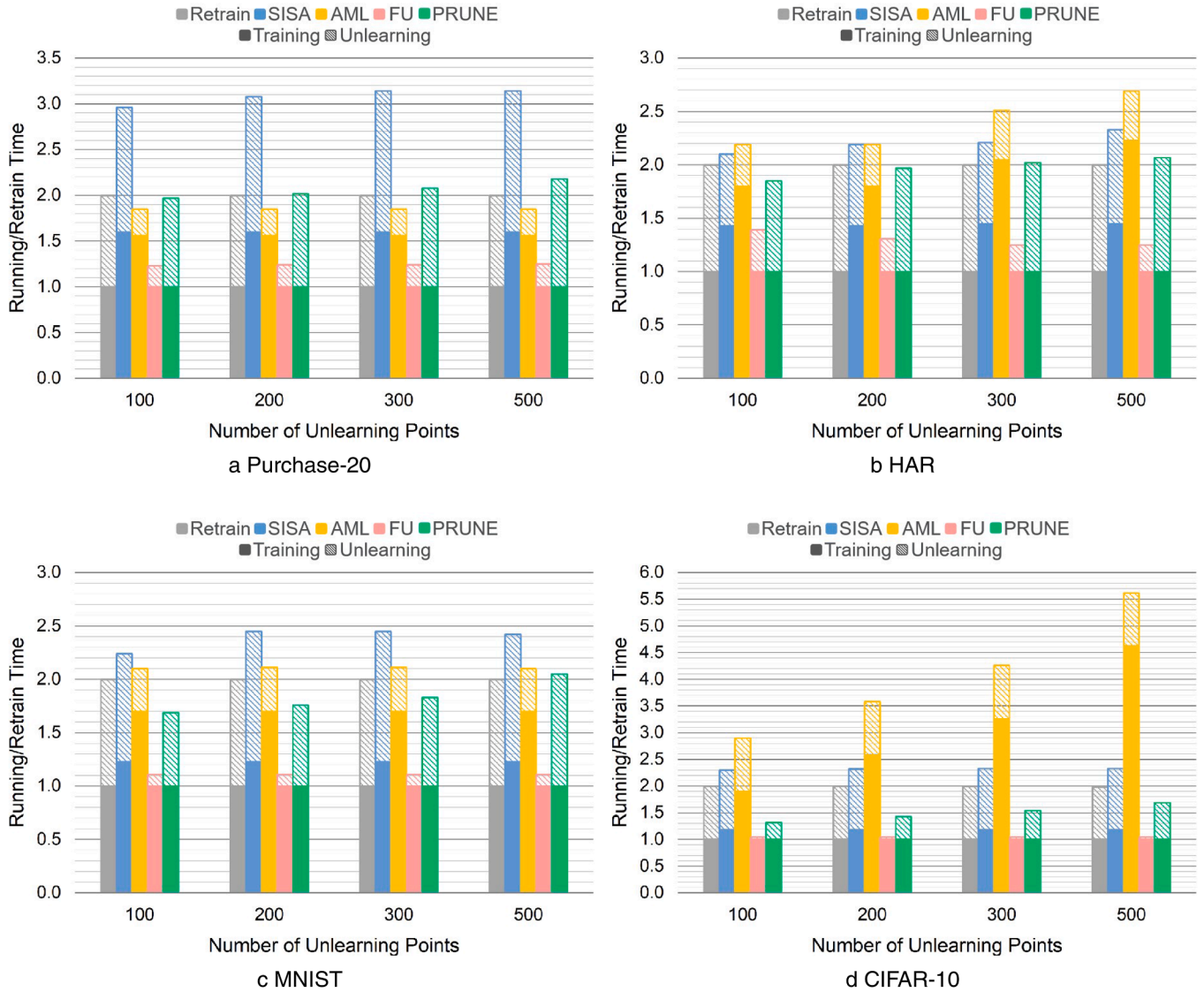


Fig. 3. The efficiency of different methods to unlearn on multiple datasets. The efficiency is measured by summing the normalized running times of the two phases. Lower values represent higher efficiency.

Remark 3. PRUNE is able to fool membership inference on unlearned data.

Remark 4. PRUNE is effective in class unlearning while not affecting other classes.

of optimized $m(x)$ at each iteration, thus influencing the convergence and time complexity of our algorithm. Fig. 4 shows the accuracy of the model on D_U with the number of iterations when different K are chosen heuristically. It can be observed that as K increases, the number of iterations required by the algorithm decreases. However, it should be noted that the increase of K makes the computational cost of each iteration higher. Ignoring other influencing factors, the time complexity of our algorithm can be briefly expressed as $\mathcal{O}(K \times Iteration)$. Therefore, for the image datasets, our method has the lowest computational cost when $K = 2$. This is due to the fact that fewer centroids will allow

Table 5

Results of the entire class unlearning on the datasets using PRUNE.

Acc(%)	Phase	Purchase-20	HAR	MNIST	CIFAR-10
A_u	before	98.14	97.51	98.95	95.52
	after	0.00	0.00	0.13	0.00
$A_{tes,u}$	before	94.75	95.91	97.4	87.70
	after	7.15	1.55	17.08	9.02
A_r	before	98.40	97.61	99.80	97.72
	after	90.68	95.88	99.26	91.14
$A_{tes,r}$	before	95.31	95.80	98.08	87.95
	after	91.24	90.01	96.86	82.99

their features to represent more intra-cluster data. The optimized $m(x)$ affects more data points as well. For the Purchase-20 dataset, our algorithm is most efficient for $K = 3$. One possible reason is that the closer distance between data points exhibiting high similarity in this dataset (binary dataset). For the HAR dataset, the choice of K does not reflect a significant difference in time complexity. For the first three datasets with the same model structure, when the value of K is the same, the convergence speed of PRUNE is positively correlated with the number of categories in the dataset. This is due to the fact that the algorithm

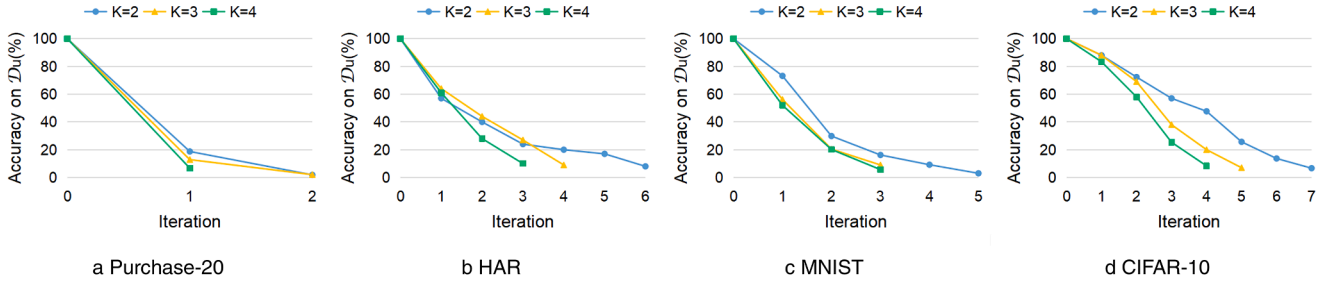


Fig. 4. The convergence of PRUNE w.r.t the number of centroids K ($|D_U| = 100$).

is optimized randomly for each data point to be withdrawn in the output space of the model, and the more labels means that it is possible to avoid the point being repeatedly replaced with the correct label when operating at multiple points. However, for the CIFAR-10 dataset with a more structurally complex model, PRUNE requires more iterations to reach the convergence condition. As demonstrated by the experiments in Section 5.3, the time cost remains manageable.

Varying the scale of D_U

The HAR dataset and the MNIST dataset are chosen for discussion here. Fig. 5 shows the accuracy of the model on D_U with the number of iterations when different scales of D_U are taken. The results of the other two datasets are shown in Li (2024). They converge in different numbers of iterations, but the conclusions they reach are consistent. We can observe that PRUNE converges quickly no matter how the scale of D_U varies. This proves the superiority of our algorithm on multipoint withdrawal. However, as the size of D_U increases, there is a tendency for PRUNE to converge faster. This trend is most obvious on the HAR dataset. When $|D_U| = 500$, the average accuracy of the model on D_U after 3 iterations is 3.1%. It is 1/3 of the accuracy of the model when $|D_U| = 100$. As discussed earlier, our algorithm has more difficulty converging on datasets with fewer categories (the conclusion can also be obtained by comparing the decreasing trend in Fig. 5a and b). Increasing the size of D_U alleviates this problem to some extent. A larger D_U scale implies a broader range of data points to be erased, enhancing the representativeness of our clustering selection before each iteration.

Remark 5. In our evaluation, PRUNE converges reasonably fast and only needs few clusters.

6. Discussion

Repair methods. Our framework focuses on provable patch based repair methods due to its certifiable guarantee, easily measurable unlearning effect, flexibility and reasonable efficiency. Repair methods that do

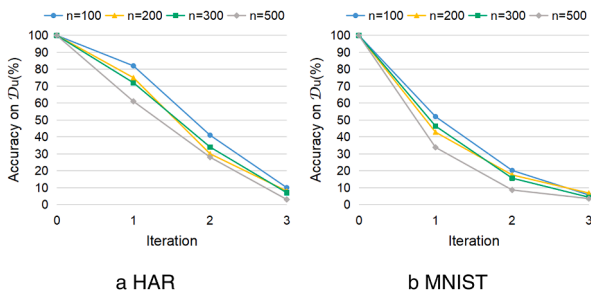


Fig. 5. The convergence of PRUNE w.r.t the scale of D_U ($K = 4$).

not have certifiable guarantee tend to be more efficient which might be more favorable in the interest of model owners. That said, as a starting point, PRUNE aims to provide a new perspective and an easily extensible repair framework which has the potential to solve diverse machine unlearning scenarios in the future.

Data type. Currently, the data types we address do not contain text data yet. With the rise of large language models, there is significantly growing concern on the sensitive information privacy risk of the text data for training, where unlearning will no doubt play a key role in protecting the data holder’s “right to be forgotten”. We are planning our next work to extend PRUNE to the scenario of removing specific text data on generating natural language models.

Representative data selection. The clustering process for the CIFAR-10 dataset is performed after extracting features using convolutional layers. This representative data selection method offers both effectiveness and speed improvements. In practice, users of PRUNE can use clustering based on different levels of pre-processed data (instead of the original data) to improve PRUNE’s performance.

Backdoor attack defense. Backdoor attacks manipulate the output of the model by injecting hidden patterns or triggers during the training phase. It can compromise the performance and reliability of AI models. Data-based defense against backdoor attacks overlaps with the design goals of PRUNE when used for class unlearning. Using reverse engineering to extrapolate the triggers, it is possible to determine the fixed activation pattern on the model for each backdoor attack. By analogizing the activation mode of backdoor to the data to be withdrawn, PRUNE has the potential to be adapted to disable these backdoor attacks without degrading the overall performance of the model much.

7. Related work

7.1. Machine unlearning

Considering the privacy issues involved in machine learning, Cao and Yang (2015) first proposed the concept of “machine unlearning”. They transformed the machine learning algorithm into a summation form of statistical query learning, and unlearned data by removing the relevant information of this sample in the summation. Since then, several unlearning algorithms have been studied for the model with convex loss function (Izzo et al., 2021; Neel et al., 2021; Nguyen et al., 2020). For instance, Guo et al. (2020) incorporated the idea of differential privacy (Abadi et al., 2016; Dwork, 2006) to provide a certified data removal mechanism for linear models. The mechanism is based on Newton updates to eliminate the effect of the removed data to achieve privacy guarantees.

However, the loss function of neural networks is non-convex, so the above approaches are not applicable to solve the unlearning problem of neural networks. In fact, the most naive method to achieve unlearning is to train from scratch on the dataset without the removed data. The downside of retraining is the high computational cost. Thus, Bourtole et al. (2021) proposed the SISA algorithm. They trained the data in shards to obtain multiple sub-models and recorded the parameters

of the sub-models in segments during the training process. When the data needs to be removed, retraining can start from the step before the sub-model parameters were affected by the deleted data. The idea of SISA is a general algorithm design for unlearning that conforms to the criterion of retraining and reduces the computational cost. Therefore, some researchers have extended this idea to unlearning in other areas such as recommendation system (Chen et al., 2022a) and graph neural networks (Chen et al., 2022b). However, training sub-models from scratch is also impractical for neural networks in many scenarios. Some studies turn to consider how to directly manipulate model parameters so that the model efficiently forgets specific data points. Amnesiac Machine Learning (Graves et al., 2021) saved gradient information of each training batch and quickly removed data by subtracting affected gradients. Thudi et al. (2022a) derived the unlearning error as a computational alternative to the verification error and designed a single-gradient approximate unlearning algorithm using Taylor series decomposition SGD. PUMA (Wu et al., 2022), on the other hand, focused on the overall performance of the model. It simulated the training contribution of each data point and weighted the contribution of the remaining data points to eliminate the effect of removing specific data points when there were data points to drop out of the training process. Recent works further extend post-processing based unlearning to graph neural networks. For example, Dynamic Graph Unlearning (Zhang et al., 2025) proposes a gradient transformation mechanism that directly maps unlearning requests to parameter updates in continuous-time dynamic graphs. It avoids retraining and achieves efficient unlearning on dynamic graph neural networks. GraphGuard (Wu et al., 2023a) studies data misuse detection and mitigation in GNN models deployed in MLaaS platforms. It introduces a training-data-free framework that combines membership inference based detection with synthetic graph based targeted unlearning. These approaches demonstrate that post-processing based unlearning can be generalized beyond standard classification models to more complex graph learning settings.

In all the above works, their goal is to obtain a model that is indistinguishable from the retrained model by unlearning algorithm. However, for the training of neural network models, different data can lead to similar gradient descent (Shumaylov et al., 2021). This means that the models with the same performance can be obtained even if a small fraction of data points are missing in the training set. On this basis, Thudi et al. (2022b) stated that unlearning could only be defined at the algorithmic level whether it was performed or not. So considering the practical application level, data holders have no means of supervising model owners to perform the unlearning algorithm. They can only observe the output of the model on their own data. From the data holder's angle, there are some unlearning algorithms that are oriented to the model prediction results. Tarun et al. (2023) generated the maximum error noise matrix for the entire class of data to be forgotten. The original model loses the ability to judge the entire category after learning the noise matrix quickly. Chen et al. (2023) directly modified the decision boundaries of the original model so that the model produces incorrect judgments for the data in a particular category. When the dataset to be unlearned is distributed in multiple categories, it is irrational to directly forget the whole class data. This is the reason why we propose an unlearning algorithm based on neural network repair. Our approach is more general which focuses on unlearning multiple randomly distributed data points by obfuscating the model's judgment about the dataset to be unlearned while minimizing the impact on the model performance.

7.2. Neural network repair

Existing neural network repair methods can be basically classified into three categories: retraining/fine-tuning, weight modification, and patching network. Retraining/fine-tuning starts from the data level, looking for data that are more suitable for the given task or more representative of model flaws, and using these data to retrain or fine-tune the neural network. FSGMix proposed by Ren et al. (2020) has a limited

number of error samples in the case, the training data is generated by small error samples. In addition to generating data, MODE (Ma et al., 2018) identifies the features that lead to misclassification through state differential analysis and guides the selection of existing or new samples for retraining the model. However, the reliability of this type of repair algorithms is difficult to prove. There is randomness in retraining or fine-tuning. They do not guarantee that errors can be fixed or that new errors will not be introduced. In addition, retraining can be very expensive. And if access to the original training data is required, this is not possible when the neural network is obtained from a third party or when the training data is private.

Weights modification first locates the neuron weights that are closely related to the erroneous behavior and then modifies them directly. NNrepair (Usman et al., 2021) identifies the neuron weights to be adjusted by fault localization and uses constraint solving to adjust the weights to fix the network. However, NNrepair is only applicable to a single layer, while in practice bugs may exist across layers. Sohn et al. (2022) proposed a search-based repair method, Arachne. Arachne identifies the weights associated with a specified faulty behavior and then optimizes the set of weights using the PSO algorithm. Goldberger et al. (2020) proposed a neural network repair method based on neural network verification to find the minimum layered repair for a given point. However, it can only perform single point repair and cannot effectively deal with polytope repair problems.

Unlike retraining/fine-tuning to adjust all parameters of the model and weight localization to modify some of them, patching network extends the structure of the neural network with faults for more efficient repair. For example, DeepCorrect (Borkar & Karam, 2019) evaluates the sensitivity of convolutional filters to distorted inputs. It then adds correction units at the output of the most distortion-sensitive convolutional filters, helping to restore some of the lost performance of the network by correcting the output of these filters. REASSURE (Fu & Li, 2022) is a neural network repair mechanism with soundness and completeness guarantees. The main idea of REASSURE is to generate a suitable patch network for linear regions in the presence of buggy inputs and combining it with the original network. The new model is able to perform correctly on the bug input.

8. Conclusion

In this paper, we propose a novel certifiable unlearning algorithm PRUNE to erase specific data by adding a targeted patch network to the original model. Unlike existing works, our approach eases data holders' concern by providing easily measurable forgetting effect (examining the model's prediction on the unlearned data) while not affecting the model's overall performance. Extensive experiments have demonstrated that PRUNE can efficiently unlearn multiple data points and an entire category data points. In order to support further development of certifiable machine unlearning, we make all the codes available at the public repository Li (2024).

CRedit authorship contribution statement

Xuran Li: Writing – original draft, Methodology, Investigation, Data curation, Conceptualization; **Jingyi Wang:** Writing – review & editing, Validation, Supervision, Formal analysis, Conceptualization; **Xiaohan Yuan:** Writing – review & editing, Supervision, Investigation; **Peixin Zhang:** Writing – review & editing, Methodology, Formal analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research was supported by the **Key Research and Development Program of Zhejiang Province** under Grant No. 2025C01083.

Appendix A. Intuition and visualization of the support network

For completeness, this appendix provides additional visualization of the support network defined in [Lemma 2](#). In ReLU-based neural networks, the input space is partitioned into a collection of linear regions, each corresponding to a fixed pattern of ReLU activations. Given a target input x_u , its associated linear region $S(x_u)$ can be expressed as a polytope defined by a set of linear inequalities $Ax \leq b$. Within this region, the network behaves as an affine function, while outside the region the activation pattern changes.

The support network is constructed to be active only inside $S(x_u)$. This is achieved by applying a ReLU-difference function to each constraint slack variable $y_i = b_i - a_i x$. Specifically, the term $\text{ReLU}(\lambda y_i + 1) - \text{ReLU}(\lambda y_i)$ is nonzero only when the corresponding constraint is satisfied. The parameter λ controls the sharpness of the transition at the boundary of the linear region. Larger values of λ result in a tighter approximation of a hard indicator function.

By summing the ReLU-difference terms associated with all constraints and applying a final ReLU operation, the resulting support network outputs a positive value if the input lies within $S(x_u)$ and outputs

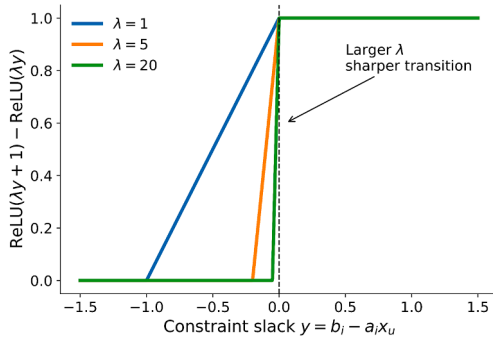


Fig. A.1. Visualization of the ReLU-difference function used in the support network. The function acts as a soft indicator of linear constraints, where positive slack values activate the support network. The hyperparameter λ controls the transition sharpness: as $\lambda \rightarrow \infty$, the function converges to a hard Heaviside step function, ensuring the patch is strictly confined to the target linear region without introducing gradient discontinuities during optimization.

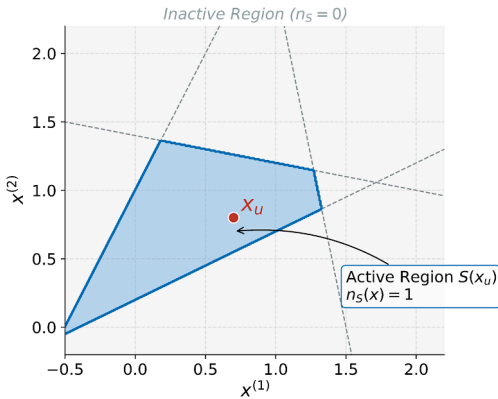


Fig. A.2. Visualizing the support network as a spatial filter. The shaded blue region represents the convex polytope $S(x_u)$ formed by intersecting linear constraints. The support network is active ($n_S = 1$) only within this specific region and strictly zero elsewhere, guaranteeing localized unlearning without side effects.

zero otherwise. As a consequence, the support network introduces localized modifications confined to the linear region of the target input and does not affect inputs outside this region.

[Fig. A.1](#) illustrates the ReLU-difference function in a one-dimensional setting, highlighting its activation behavior near the region boundary. [Fig. A.2](#) presents a two-dimensional toy example in which the activation region of the support network closely matches the underlying linear region induced by ReLU activations.

Appendix B. Membership inference attack setup

The privacy robustness of PRUNE is evaluated using the membership inference attack (MIA) framework proposed by [Yeom et al. \(2018\)](#). In this black-box, shadow-model-free setting, the adversary queries the unlearned model M_U to obtain the prediction loss $L(M_U(x), y)$ for a target sample. The core principle of this attack is that a model's training loss is generally lower than its test loss. An adversary identifies a sample as a member if its observed loss is below a specific confidence threshold τ . In our experiments, τ is fixed as the average training loss of the original model M_D . This serves as a fixed privacy baseline to evaluate whether the unlearned sample x_u has been successfully “pushed” out of the membership loss distribution. All remaining hyperparameters are kept identical to those used in the original training setting unless otherwise specified. Notably, this approach allows for a direct assessment of privacy leakage without the need for training auxiliary shadow models, as it relies on the intrinsic generalization gap of the target model. We use the recall of this inference process to quantify the extent to which PRUNE effectively erases the membership information of the unlearned data.

Appendix C. Termination and complexity analysis

In [Algorithms 1](#) and [2](#), the stopping criterion is defined as

$$1 - \frac{|D_{UR}|}{|D_U|} > \delta, \quad (\text{C.1})$$

where D_{UR} denotes the set of samples that remain unsuccessfully unlearned after the current iteration. The parameter $\delta \in (0, 1)$ represents the target unlearning degree. Specifically, the algorithm terminates once the fraction of effectively unlearned samples reaches the threshold δ . In all our experiments, we set $\delta = 0.95$, ensuring that at least 95% of the requested data points are successfully removed from the model's original predictive influence. This threshold provides a rigorous balance between near-complete unlearning and the computational efficiency of the patching process.

This termination condition is well-defined and guarantees finite convergence. Specifically, at each iteration, the update procedure either successfully removes samples from D_U or leaves the set unchanged, implying that $D_{UR} \subseteq D_U$ and that $|D_{UR}|$ is monotonically non-increasing across iterations. Consequently, the ratio $|D_{UR}|/|D_U|$ decreases monotonically, ensuring that the stopping condition will be satisfied after a finite number of iterations.

From a complexity perspective, in the worst case, each iteration removes at least one sample from D_U , leading to an upper bound of $O(|D_U|)$ iterations. In practice, however, the algorithm converges much faster. Across all evaluated settings, the number of iterations is typically fewer than five, indicating that the proposed procedure is efficient and stable in realistic scenarios.

Appendix D. Theoretical generalization

In the main text, we presented PRUNE under the assumption of CPWL structures, specifically ReLU-based networks, to derive exact certifiability. In this section, we provide formal proofs extending our framework to (1) arbitrary feature extractors including Transformer-based architectures ([Vaswani et al., 2017](#)), and (2) non-ReLU activation func-

tions such as GELU (Hendrycks, 2016) and SiLU (Jocher et al., 2021), via Lipschitz continuous approximation.

D.1. Generalization to arbitrary backbones

Proposition 1. (Structural Independence). According to Section 4.2, let a neural network be decomposed as $M(x) = M_c(M_p(x))$, where $M_p : \mathcal{X} \rightarrow \mathcal{Z}$ is a feature extractor with arbitrary architecture, and $M_c : \mathcal{Z} \rightarrow \mathcal{Y}$ is a CPWL classification head. The existence of a valid unlearning patch c_S for a target x_u depends solely on the properties of M_c and the latent vector $z_u = M_p(x_u)$, independent of the internal structure of M_p .

Proof. Recall that the PRUNE framework constructs the patch network c_S based on the feasible set $S(x_u)$ derived from the neuron activation states. Let the input to the classification head be $z \in \mathcal{Z}$, where $z = M_p(x)$. The classification head M_c is a composition of ReLU layers. For a target sample x_u , the fixed feature vector is $z_u = M_p(x_u)$.

Since M_c is CPWL, the activation pattern of z_u within M_c defines a linear polytope $S_c(z_u) \subset \mathcal{Z}$ in the latent space, characterized by:

$$S_c(z_u) = \{z \in \mathcal{Z} \mid \mathbf{A}z \leq \mathbf{b}\} \quad (\text{D.1})$$

where \mathbf{A} and \mathbf{b} are determined specifically by the weights of M_c and the sign configuration of z_u .

The support network $n_S(z)$ is constructed to be active if and only if $z \in S_c(z_u)$. The final unlearned model is:

$$M_U(x) = (M_c + c_S)(M_p(x)) \quad (\text{D.2})$$

Since M_p is frozen during the patch construction, it acts as a deterministic mapping from \mathcal{X} to \mathcal{Z} . The verification of conditions 1) and 2) in Theorem 1 reduces to:

- For x_u : $M_p(x_u) = z_u \in S_c(z_u) \implies n_S(z_u) = 1$. The patch is active, achieving forgetting.
- For $x \neq x_u$: If $M_p(x) \notin S_c(z_u)$, then $n_S(M_p(x)) = 0$. The patch is inactive, preserving performance.

Crucially, this derivation imposes no constraints on the Jacobian or continuity of M_p . Thus, M_p can be a Vision Transformer, a BERT-style encoder, or any non-linear block, provided the final head M_c is ReLU-based. \square

D.2. Extension to Non-ReLU activations

Modern architectures often use smooth activations (e.g., GELU) which are not strictly CPWL. We show that PRUNE provides an ϵ -bounded unlearning guarantee for such models.

Definition 2. (ϵ -CPWL approximation). A smooth activation function $\sigma(x)$ is ϵ -approximable if there exists a CPWL function $\tilde{\sigma}(x)$ such that $\sup_x |\sigma(x) - \tilde{\sigma}(x)| \leq \epsilon$. Note that common functions like GELU and SiLU are Lipschitz continuous and can be approximated by Piecewise Linear functions to arbitrary precision.

Proposition 2. (Bounded side-effects). Let M be a network with smooth activations and \tilde{M} be its ϵ -CPWL approximation. Let c_S be the unlearning patch derived exactly for \tilde{M} . Applying c_S to the original model M results in an unlearned model $M_U = M + c_S$ with side effects bounded by $\mathcal{O}(\epsilon)$.

Proof. Let D be the depth of the network and K be the Lipschitz constant of the layers. The difference between the original model output and the approximated model output is bounded by:

$$\sup_x |M(x) - \tilde{M}(x)| \leq \phi(\epsilon, D, K) \quad (\text{D.3})$$

where ϕ is a function that scales linearly with ϵ .

We construct the patch c_S based on \tilde{M} , such that $\tilde{M}_U = \tilde{M} + c_S$ achieves exact unlearning on \tilde{M} . Now we apply this patch to M : $M_U(x) = M(x) + c_S(x)$.

For a non-target data point x_{retain} , ideally, we require $M_U(x_{retain}) = M(x_{retain})$. The deviation is:

$$\begin{aligned} \text{Error} &= |M_U(x_{retain}) - M(x_{retain})| \\ &= |(M(x_{retain}) + c_S(x_{retain})) - M(x_{retain})| \\ &= |c_S(x_{retain})| \end{aligned} \quad (\text{D.4})$$

Recall that $c_S(x)$ is designed to be zero when $\tilde{M}(x)$ falls outside the feasible set $S_{\tilde{M}}(x_u)$. However, due to the approximation error, the activation path of x_{retain} in M might slightly differ from \tilde{M} .

If x_{retain} is sufficiently far from the boundary of $S(x_u)$ (margin $> \phi$), then $c_S(x_{retain}) = 0$ exactly. In the worst-case scenario where the approximation shift pushes x_{retain} into the support region of the CPWL approximation, the magnitude of the patch output is bounded by the Lipschitz continuity of the support network n_S . Since the region volume scales with λ , for a sufficiently precise approximation ($\epsilon \rightarrow 0$), the probability of x_{retain} erroneously triggering the patch is bounded and vanishes as the approximation becomes arbitrarily precise.

Thus, PRUNE guarantees exact unlearning on the ϵ -proxy model \tilde{M} , which translates to approximate unlearning on M with performance deviations bounded by the approximation error. \square

Appendix E. Experiment extension

To thoroughly evaluate the scalability and architectural robustness of the PRUNE framework, we extend our experimental evaluation to the vision transformer architecture using the CIFAR-100 dataset (Krizhevsky, 2009). We utilize the `vit_tiny_patch16_224` backbone (Wightman, 2019), where the original classification head is replaced with a MLP consisting of two hidden layers with 1024 units each. All CIFAR-100 images are dynamically upsampled to 224×224 pixels to satisfy the patch embedding requirements of the transformer architecture. We compare PRUNE against several rigorous baselines to establish a comprehensive performance benchmark. Retraining, SISA and AML are implemented here exactly as described in Section 5.1. Additionally, Gated Representation UNlearning (Ren et al., 2025) is included as a modern representation-level intervention baseline, specifically targeting the latent features within the later transformer blocks of the ViT backbone.

The experimental results, summarized in Table E.1, demonstrate the definitive superiority of PRUNE in maintaining model stability while ensuring complete erasure of target data. A critical observation is that PRUNE consistently maintains a 0.00% change in both test set accuracy ΔA_{res} and rest set accuracy ΔA_{res} across all unlearning scales, ranging from $D_U = 100$ to 500. This signifies that the patching mechanism provides absolute isolation, preventing any collateral damage to the model's

Table E.1
Performance comparison when different numbers of data points are unlearned on CIFAR-100.

D_U	Acc(%)	Retrain	SISA	AML	GRUN	PRUNE
0	A_{res}	76.36	69.62	74.84	69.69	69.69
	A_{res}	94.08	76.62	87.21	98.76	98.76
	ΔA_{res}	0.23	0.17	-1.25	14.60	0.00
100	ΔA_{res}	0.57	0.07	-4.02	26.67	0.00
	ΔA_u	21.60	7.63	3.43	89.52	97.40
	ΔA_{res}	-0.15	-0.78	1.23	21.75	0.00
200	ΔA_{res}	0.08	-0.54	1.09	37.22	0.00
	ΔA_u	17.17	6.67	9.67	85.98	98.32
	ΔA_{res}	-0.87	0.98	3.49	26.68	0.00
300	ΔA_{res}	-0.45	0.56	6.13	47.03	0.00
	ΔA_u	19.34	4.33	11.83	89.01	98.11
	ΔA_{res}	0.30	2.99	12.49	29.36	0.00
500	ΔA_{res}	-0.59	1.97	20.26	48.66	0.00
	ΔA_u	19.23	5.80	25.25	78.44	97.87

Table E.2

The efficiency of different methods to unlearn on CIFAR-100. The efficiency is measured by summing the normalized running times of the two phases.

D_U	Retrain		SISA		AML		GRUN		PRUNE	
	train	unlearn	train	unlearn	train	unlearn	train	unlearn	train	unlearn
100	1.00	1.00	0.91	0.91	1.15	0.23	1.00	3.03	1.00	1.57
200	1.00	1.00	0.91	0.91	1.16	0.23	1.00	2.68	1.00	2.45
300	1.00	1.00	0.91	0.91	1.16	0.23	1.00	2.26	1.00	3.17
500	1.00	0.99	0.91	0.91	1.16	0.23	1.00	1.89	1.00	5.08

non-target data. In contrast, the modern intervention method GRUN exhibits substantial performance degradation, with ΔA_{tes} dropping by as much as 29.36%. This suggests that gradient-based representation adjustments in complex architectures like ViT can inadvertently corrupt essential features required for general classification tasks. The slight accuracy improvement of Retraining, SISA, and AML on the test set and the remaining dataset after unlearning is due to the fact that these methods involve retraining or fine-tuning during the unlearning phase, which can cause minor fluctuations in performance. This also implies that their side effects are unavoidable. Furthermore, PRUNE achieves an unlearning efficacy consistently exceeding 97.40%, which is significantly higher than all other baselines. Notably, the Retrain baseline only reaches an efficacy of approximately 20%, indicating that simple head-retraining is insufficient to purge features that are deeply embedded during the backbone's pre-training or fine-tuning phases. While AML shows a gradual increase in efficacy as the unlearning scale grows, its performance peaks at only 25.25%, further highlighting the precision of the PRUNE approach. Overall, these findings confirm that PRUNE effectively generalizes to self-attention-based architectures and manages high-dimensional feature spaces with far greater precision and efficiency than traditional exact or approximate unlearning techniques.

The computational efficiency of the evaluated methods is summarized in Table E.2. The metrics reveal that AML is the most time-efficient baseline, maintaining a normalized unlearning cost of approximately 0.23. For GRUN, the unlearning time reduces from 3.03 to 1.89 as $|D_U|$ increases, which is attributable to the reduced number of iterations required on the shrinking remaining dataset. While the unlearning cost of PRUNE scales from 1.57 to 5.08 as the unlearning size expands, this trend reflects the necessary computational complexity required to calculate exact patches. This investment directly facilitates the superior precision and zero side effects on non-target data observed in Table E.1.

References

- Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., & Zhang, L. (2016). Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security* (pp. 308–318).
- Ben-Tal, A., El Ghaoui, L., & Nemirovski, A. (2009). *Robust optimization* (vol. 28). Princeton university press.
- Borkar, T. S., & Karam, L. J. (2019). DeepCorrect: Correcting DNN models against image distortions. *IEEE Transactions on Image Processing*, 28(12), 6022–6034.
- Bourtoule, L., Chandrasekaran, V., Choquette-Choo, C. A., Jia, H., Travers, A., Zhang, B., Lie, D., & Papernot, N. (2021). Machine unlearning. In *2021 IEEE Symposium on security and privacy (SP)* (pp. 141–159). IEEE.
- Bulbul, E., Cetin, A., & Dogru, I. A. (2018). Human activity recognition using smartphones. In *2018 2nd international symposium on multidisciplinary studies and innovative technologies (imsit)* (pp. 1–6). IEEE.
- Cao, Y., & Yang, J. (2015). Towards making systems forget with machine unlearning. In *2015 IEEE Symposium on security and privacy* (pp. 463–480). IEEE.
- Chen, C., Sun, F., Zhang, M., & Ding, B. (2022a). Recommendation unlearning. In *Proceedings of the ACM web conference 2022* (pp. 2768–2777).
- Chen, M., Gao, W., Liu, G., Peng, K., & Wang, C. (2023). Boundary unlearning. arXiv preprint arXiv:2303.11570.
- Chen, M., Zhang, Z., Wang, T., Backes, M., Humbert, M., & Zhang, Y. (2022b). Graph unlearning. In *Proceedings of the 2022 ACM SIGSAC conference on computer and communications security* (pp. 499–513).
- Chien, E., Pan, C., & Milenkovic, O. (2023). Efficient model updates for approximate unlearning of graph-structured data. In *International conference on learning representations*.
- Dong, G., Sun, J., Wang, X., Wang, X., & Dai, T. (2021). Towards repairing neural networks correctly. In *2021 IEEE 21st international conference on software quality, reliability and security (QRS)* (pp. 714–725). IEEE.
- Dwork, C. (2006). Differential privacy. In *Automata, languages and programming: 33rd international colloquium, ICALP 2006, Venice, Italy, July 10–14, 2006, proceedings, part II 33* (pp. 1–12). Springer.
- European Union (2016). Regulation (EU) 2016/679 of the European Parliament and of the Council (General Data Protection Regulation). *Official Journal of the European Union*, L 119, 1–119.
- Foster, J., Schoepf, S., & Brintrup, A. (2024). Fast machine unlearning without retraining through selective synaptic dampening. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 12043–12051). (vol. 38).
- Fu, F., & Li, W. (2022). Sound and complete neural network repair with minimality and locality guarantees. In *International conference on learning representations*.
- Gao, J., Garg, S., Mahmood, M., & Vasudevan, P. N. (2022). Deletion inference, reconstruction, and compliance in machine (un) learning. *Proceedings on Privacy Enhancing Technologies*, 3, 415–436.
- Ginart, A. A., Guan, M. Y., Valiant, G., & Zou, J. (2019). Making AI forget you: Data deletion in machine learning. In *Proceedings of the 33rd international conference on neural information processing systems* (pp. 3518–3531).
- Goldberger, B., Katz, G., Adi, Y., & Keshet, J. (2020). Minimal modifications of deep neural networks using verification. *EPiC Series in Computing*, 73, 260–278.
- Graves, L., Nagisetty, V., & Ganesh, V. (2021). Amnesiac machine learning. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 11516–11524). (vol. 35).
- Gu, T., Dolan-Gavitt, B., & Garg, S. (2017). BadNets: Identifying vulnerabilities in the machine learning model supply chain. arXiv preprint arXiv:1708.06733.
- Guo, C., Goldstein, T., Hannun, A., & Van Der Maaten, L. (2020). Certified data removal from machine learning models. In *International conference on machine learning* (pp. 3832–3842). PMLR.
- Hendrycks, D. (2016). Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415.
- Hu, X., Li, D., Hu, B., Zheng, Z., Liu, Z., & Zhang, M. (2024). Separate the wheat from the chaff: Model deficiency unlearning via parameter-efficient module operation. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 18252–18260). (vol. 38).
- Izzo, Z., Smart, M. A., Chaudhuri, K., & Zou, J. (2021). Approximate data deletion from machine learning models. In *International conference on artificial intelligence and statistics* (pp. 2008–2016). PMLR.
- Jocher, G. R., Stoken, A., Borovec, J., NanoCode, ChristopherSTAN, Changyu, L., Laughing, tkianai, yxNONG, Hogan, A., lorenzomamma, AlexWang, Chaurasia, A., Diacontu, L., Marc, wanghaoyang, ah, m., Doug, Durgesh, Ingham, F., Frederik, Guilhen, Colmagro, A., Ye, H., Jacobsolawetz, Poznanski, J., Fang, J., Kim, J., Doan, K. N., & Yu, L. (2021). Ultralytics/YOLOv5: v4.0 - nn.silu activations, weights & biases logging, pytorch hub integration. <https://api.semanticscholar.org/CorpusID:244999743>.
- Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Technical Report.
- Krizhevsky, A. et al. (2009). Learning multiple layers of features from tiny images. <https://www.cs.toronto.edu/kriz/learning-features-2009-TR.pdf>.
- LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324.
- Lee, G.-H., Alvarez-Melis, D., & Jaakkola, T. S. (2019). Towards robust, locally linear deep networks. In *International conference on learning representations*.
- Li, X. (2024). The repository of code and data to support patching based repair framework for certifiable unlearning. <https://github.com/dummyPRUNE/PRUNE2024>.
- Li, X., Wang, J., Yuan, X., & Zhang, P. (2025). PRUNE: A patching based repair framework for certifiable unlearning of neural networks. In *2025 China automation congress (CAC)*. IEEE.
- Li, Y., Wang, C.-H., & Cheng, G. (2021). Online forgetting process for linear regression models. In *International conference on artificial intelligence and statistics* (pp. 217–225). PMLR.
- Liu, Y., Xu, L., Yuan, X., Wang, C., & Li, B. (2022). The right to be forgotten in federated learning: An efficient realization with rapid retraining. In *IEEE infocom 2022-IEEE conference on computer communications* (pp. 1749–1758). IEEE.
- Ma, J., Yang, P., Wang, J., Sun, Y., Huang, C.-C., & Wang, Z. (2024). VeRe: Verification guided synthesis for repairing deep neural networks. In *Proceedings of the 46th IEEE/ACM international conference on software engineering* (pp. 1–13).
- Ma, S., Liu, Y., Lee, W.-C., Zhang, X., & Grama, A. (2018). MODE: Automated neural network model debugging via state differential analysis and input selection. In *Proceedings of the 2018 26th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering* (pp. 175–186).
- Neel, S., Roth, A., & Sharifi-Malvajerdi, S. (2021). Descent-to-delete: Gradient-based methods for machine unlearning. In *Algorithmic learning theory* (pp. 931–962). PMLR.
- Nguyen, Q. P., Kian, B., Low, H., & Jalliet, P. (2020). Variational Bayesian unlearning. In *Proceedings of the 34th international conference on neural information processing systems* (pp. 16025–16036).

- Ren, J., Dai, Z., Tang, X., Liu, H., Zeng, J., Li, Z., Goutam, R., Wang, S., Xing, Y., & He, Q. (2025). A general framework to enhance fine-tuning-based LLM unlearning. arXiv preprint arXiv:2502.17823.
- Ren, X., Yu, B., Qi, H., Juefei-Xu, F., Li, Z., Xue, W., Ma, L., & Zhao, J. (2020). Few-shot guided mix for DNN repairing. In *2020 IEEE International conference on software maintenance and evolution (ICSME)* (pp. 717–721). IEEE.
- Samek, W., Wiegand, T., & Müller, K.-R. (2017). Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *ITU Journal: ICT Discoveries - Special Issue 1 - The Impact of Artificial Intelligence (AI) on Communication Networks and Services, 1*, 1–10.
- Shibata, T., Irie, G., Ikami, D., & Mitsuzumi, Y. (2021). Learning with selective forgetting. In *30th international joint conference on artificial intelligence, IJCAI 2021* (pp. 989–996). International Joint Conferences on Artificial Intelligence.
- Shokri, R., Stronati, M., Song, C., & Shmatikov, V. (2017). Membership inference attacks against machine learning models. In *2017 IEEE Symposium on security and privacy (SP)* (pp. 3–18). IEEE.
- Shumaylov, Z., Kazhdan, D., Zhao, Y., Papernot, N., Erdogdu, M. A., & Anderson, R. J. (2021). Manipulating SGD with data ordering attacks. *Advances in Neural Information Processing Systems, 34*, 18021–18032.
- Sohn, J., Kang, S., & Yoo, S. (2023). Arachne: Search based repair of deep neural networks. *ACM Transactions on Software Engineering and Methodology, 32*(4), 1–26.
- Sotoudeh, M., & Thakur, A. V. (2019). Correcting deep neural networks with small, generalizing patches. In *NeurIPS 2019 workshop on safety and robustness in decision making*.
- Sotoudeh, M., & Thakur, A. V. (2021). Provable repair of deep neural networks. In *Proceedings of the 42nd ACM SIGPLAN international conference on programming language design and implementation* (pp. 588–603).
- Sun, B., Sun, J., Pham, L. H., & Shi, J. (2022). Causality-based neural network repair. In *Proceedings of the 44th international conference on software engineering* (pp. 338–349).
- Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2014). Intriguing properties of neural networks. In *2nd international conference on learning representations, ICLR 2014*.
- Tarun, A. K., Chundawat, V. S., Mandal, M., & Kankanhalli, M. (2024). Fast yet effective machine unlearning. *IEEE Transactions on Neural Networks and Learning Systems, 35*(9), 13046–13055.
- Thudi, A., Deza, G., Chandrasekaran, V., & Papernot, N. (2022a). Unrolling SGD: Understanding factors influencing machine unlearning. In *2022 IEEE 7th European symposium on security and privacy (euros&p)* (pp. 303–319). IEEE.
- Thudi, A., Jia, H., Shumailov, I., & Papernot, N. (2022b). On the necessity of auditable algorithmic definitions for machine unlearning. In *31st USENIX security symposium (USENIX security 22)* (pp. 4007–4022).
- Usman, M., Gopinath, D., Sun, Y., Noller, Y., & Păsăreanu, C. S. (2021). NN repair: Constraint-based repair of neural network classifiers. In *Computer aided verification: 33rd international conference, CAV 2021, virtual event, july 20–23, 2021, proceedings, part i 33* (pp. 3–25). Springer.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*. Curran Associates, Inc. (vol. 30).
- Warnecke, A., Pirch, L., Wressnegger, C., & Rieck, K. (2023). Machine unlearning of features and labels. In *Proc. of the 30th network and distributed system security (NDSS)*.
- Wightman, R. (2019). Pytorch image models. <https://doi.org/10.5281/zenodo.4414861>
- Wu, B., Zhang, H., Yang, X., Wang, S., Xue, M., Pan, S., & Yuan, X. (2023a). GraphGuard: Detecting and counteracting training data misuse in graph neural networks. arXiv preprint arXiv:2312.07861.
- Wu, G., Hashemi, M., & Srinivasa, C. (2022). PUMA: Performance unchanged model augmentation for training data removal. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 8675–8682). (vol. 36).
- Wu, J., Yang, Y., Qian, Y., Sui, Y., Wang, X., & He, X. (2023b). GIF: A general graph unlearning strategy via influence function. In *Proceedings of the ACM web conference 2023* (pp. 651–661).
- Yan, H., Li, X., Guo, Z., Li, H., Li, F., & Lin, X. (2022). ARCANE: An efficient architecture for exact machine unlearning. In *Proceedings of the thirty-first international joint conference on artificial intelligence, IJCAI-22* (pp. 4006–4013).
- Yeom, S., Giacomelli, I., Fredrikson, M., & Jha, S. (2018). Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st computer security foundations symposium (CSF)* (pp. 268–282). IEEE.
- Zhang, H., Weng, T.-W., Chen, P.-Y., Hsieh, C.-J., & Daniel, L. (2018). Efficient neural network robustness certification with general activation functions. *Advances in Neural Information Processing Systems, 31*, 4944–4953.
- Zhang, H., Wu, B., Yang, X., Yuan, X., Liu, X., & Yi, X. (2025). Dynamic graph unlearning: A general and efficient post-processing method via gradient transformation. In *Proceedings of the ACM on web conference 2025* (pp. 931–944).
- Zhang, Z., Zhou, Y., Zhao, X., Che, T., & Lyu, L. (2022). Prompt certified machine unlearning with randomized gradient smoothing and quantization. *Advances in Neural Information Processing Systems, 35*, 13433–13455.
- Zhou, J., Li, H., Liao, X., Zhang, B., He, W., Li, Z., Zhou, L., & Gao, X. (2023). Audit to forget: A unified method to revoke patients' private data in intelligent healthcare. *14*(1), 6255.
- Zhou, Y., Zheng, D., Mo, Q., Lu, R., Lin, K.-Y., & Zheng, W.-S. (2025). Decoupled distillation to erase: A general unlearning method for any class-centric tasks. In *Proceedings of the computer vision and pattern recognition conference* (pp. 20350–20359).