

PRUNE: A Patching Based Repair Framework for Certifiable Unlearning of Neural Networks

Xuran Li, Jingyi Wang, Xiaohan Yuan, Peixin Zhang, Zhan Qin, *Senior Member, IEEE*, Zhibo Wang, *Senior Member, IEEE*, and Kui Ren, *Fellow, IEEE*

Abstract—It is often desirable to remove (a.k.a. unlearn) a specific part of the training data from a trained neural network model. A typical application scenario is to protect the data holder’s right to be forgotten, which has been promoted by many recent regulation rules. Existing unlearning methods involve training alternative models with remaining data, which may be costly and challenging to verify from the data holder or a third-party auditor’s perspective. In this work, we provide a new angle and propose a novel unlearning approach by imposing carefully crafted ‘patch’ on the original neural network to achieve targeted ‘forgetting’ of the requested data to delete. Specifically, inspired by the research line of *neural network repair*, we propose to strategically seek a lightweight minimum ‘patch’ for unlearning a given data point with certifiable guarantee. Furthermore, to unlearn a considerable amount of data points (or an entire class), we propose to iteratively select a small subset of representative data points to unlearn, which achieves the effect of unlearning the whole set. Extensive experiments on multiple categorical datasets demonstrates our approach’s effectiveness, achieving measurable unlearning while preserving the model’s performance and being competitive in efficiency and memory consumption compared to various baseline methods.

Index Terms—Machine Learning, Machine Unlearning, Privacy Leakage, Data Privacy.

I. 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 [1], 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 [2].

To support the request of data erasure (or removal), the study on *machine unlearning* [3]–[10] 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 Arguably, the most intuitive way to unlearn is to retrain an alternative model M_r after removing the data to be withdrawn from the training set¹.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 [6], [10]–[14]. They often perform additional operations during the model training phase to reduce the cost of retraining by, for example, either slicing the data [6] or segmenting the training [9]. Approximate unlearning [2], [4], [15]–[20], 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 [16], [21], [22].

However, as pointed out by [23], one limitation of these unlearning recipe is that *it remains difficult to erase the data holder’s privacy concern or convince a third-party auditor as the retrained model may 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 [24], [25]. 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 [26], [27] have been conducted with such two sides of the coin in mind. But their granularity is too coarse-grained which focused 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 data just for a partial delete request.

Xuran Li, Jingyi Wang, Xiaohan Yuan, Zhan Qin, Zhibo Wang and Kui Ren are with the Zhejiang University, Hangzhou, Zhejiang 310007, China. E-mail: xuranli1005@zju.edu.cn, wangjyee@zju.edu.cn, xiaohanyuan0@gmail.com, qinzhan@zju.edu.cn, zhibowang@zju.edu.cn, kuiren@zju.edu.cn.

Peixin Zhang is with the Singapore Management University, Singapore 188065. E-mail: pxzhang94@gmail.com.

¹We 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.

In this work, we take a new angle and propose a novel machine unlearning mechanism 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 coincide with the *neural network repair problem*, whose goal is to *correct the model's misjudgment on some erroneous data while minimizing the impact on other data*. This motivates us to connect machine unlearning with neural network repair [28]–[31] 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. 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.

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 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 separately, 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.

II. PRELIMINARIES

A. 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 $\mathcal{D} = \{x_i, y_i\}_{i=1, \dots, n}$ 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. $M_{\mathcal{D}}$ is a DNN model trained on the dataset \mathcal{D} . It makes judgment about the label of $x_i : \arg \max_{l \in \mathcal{Y}} M_{\mathcal{D}}^l(x_i)$, where $M_{\mathcal{D}}^l(x)$ denotes the output probability that model $M_{\mathcal{D}}$ considers the label of x to be l . When there are requests for erasure, $\mathcal{D}_U = \{x_u, y_u\}_{u=1, \dots, r}$ denotes the set of data points to be unlearned and $\mathcal{D}_U \subset \mathcal{D}$. $\mathcal{D}_R = \mathcal{D} / \mathcal{D}_U$ is the set of remaining data points. M_U is the model after the execution of the unlearning algorithm.

B. 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 [33], [34]. 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 [35] is more challenging which makes repair efforts difficult in general.

Existing repair methods mainly include retraining/fine-tuning, modifying parameters [36], [37] and patching network [30], [38], [39]. 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 $x_r \in X_R$,

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

²The code and data are released at [32]

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

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

where \mathcal{D} is the data set that does not need to be repaired.

C. 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 makes the correct judgement on his/her data. This objective can be formalized as for any $x_u \in \mathcal{D}_U$,

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

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

Besides the unlearning objective in Equation (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 \mathcal{D}_R . Thus, the other objective in Equation (4) is to make the performance change on the remaining data as small as possible. It is formulated as

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

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.

III. THREAT MODEL

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 Figure 1. The model owner uses the data from the data holders for training the model. According to the GDPR [1] 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.

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 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.

For this threat model, we default to a possible dishonest model owner. In this scenario, retraining is not the most reasonable strategy to unlearn - even ignoring the high computational cost, due to the inherent difficulty in measuring the impact of the model's generalization ability [23]. The models obtained by retraining after removing the data to be unlearned often do not show differences in their output significantly. The model owner can simply claim that they have performed the unlearning operation while they actually did not. If the auditors use the retrained model as a golden-standard judgment, they are unlikely to draw reliable conclusions and convince the data holders just by comparing the outputs. This undoubtedly undermines the “right to be forgotten” of each data holder and can not erase their privacy concern.

IV. 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.

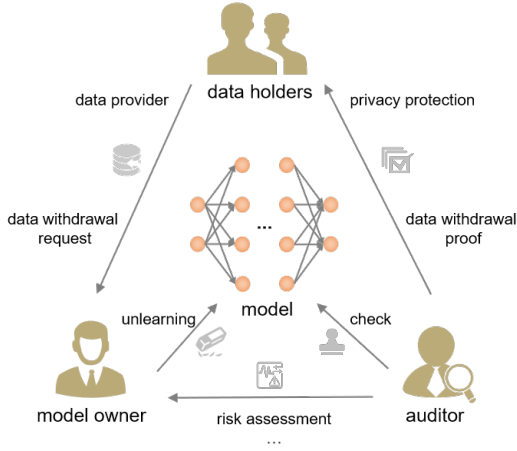


Fig. 1: There are three parties involved in the threat model: the data holders, the model owner, and the auditor.

A. Unlearning Properties

A qualified machine unlearning mechanism should meet the following properties.

1) *Utility on \mathcal{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.

B. The PRUNE Framework

In the following, we present our *Patching based Repair* framework for certifiable machine UNlearning (PRUNE) in details. As shown in Figure 2, the key idea of PRUNE is to generate a targeted “patch” network on the original model M_D training on \mathcal{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 [32].

Theorem 1: Given a neural network model $M_D = M_p \oplus M_c$ trained on \mathcal{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 $x \in \mathcal{D}/x_u$, $M_U(x) = M_D(x)$.

Next, we introduce how the patch network c_S is constructed. In general, a *patch network* consists of two parts: a *confusion sub-network* 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 Equation 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 [40].

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 $\bar{x}_u \in \mathcal{X}$,

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

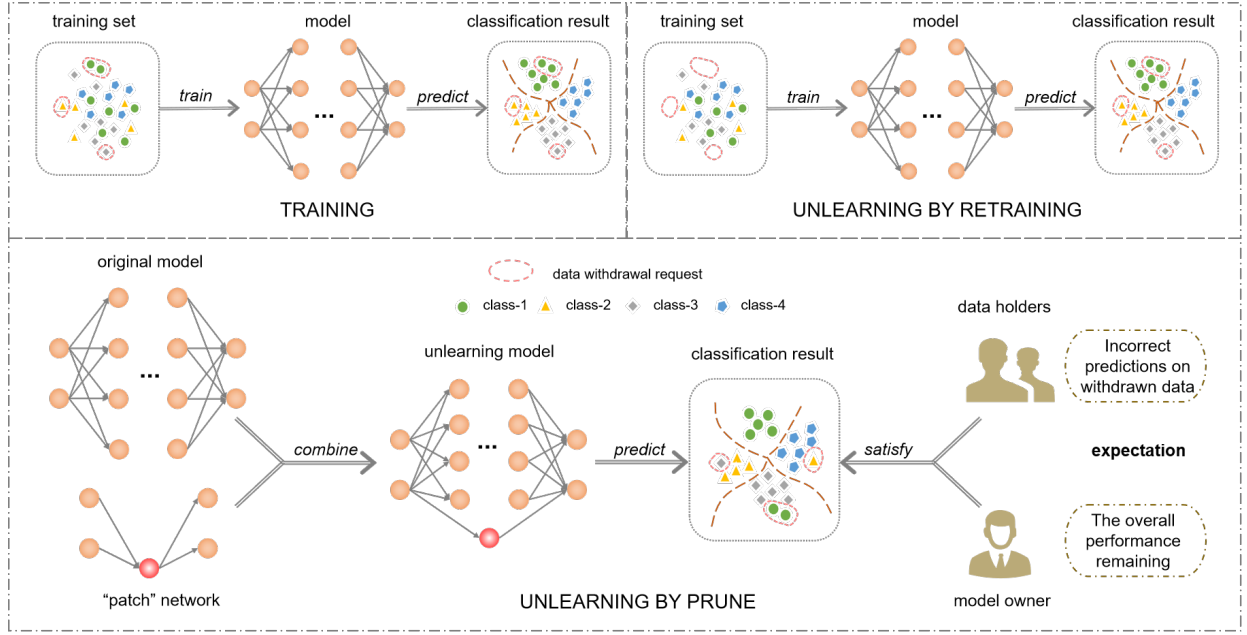


Fig. 2: The framework of PRUNE. When data withdrawal requests occur, PRUNE reduces the model's ability to judge these specific data points by generating a patch network. This mechanism solves the problem that the generalization ability of the model makes the unlearning process impossible to verify intuitively. 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.

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 Equation 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 [39].

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 $n(x_u)$.

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

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

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

Equation 7 can be converted to a linear programming (LP) problem by enumerating the vertices and be solved by robust optimization [41].

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. 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 Equation 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

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

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 Equation 7. We feed $\{x_u^k\}_{u=1,\dots,r'}$ into the temporary network $M_{\mathcal{D}} + m_k$ 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 Equation 6. If the output of this point has not changed, it will be recorded into the remaining unlearning dataset \mathcal{D}_{UR} for a new round of iteration. After traversing all points in \mathcal{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 \mathcal{D}_U have been optimized m_k , tested, generated $\{n_u^k\}_{u=1 \rightarrow r'}$ 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 $1 - \mathcal{D}_{UR}/\mathcal{D}_U$. When it is higher than the required 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.

Theorem 2: Given a neural network model $M_{\mathcal{D}} = M_p \oplus M_c$ trained on \mathcal{D} , a set of data $\mathcal{D}_U \subset \mathcal{D}$ to unlearn and a desired unlearning degree δ on \mathcal{D}_U , it is guaranteed that we can construct a series of patch network $\{c_k\}_{k=1 \rightarrow K}$ for M_c and obtain an unlearned model $M_U = M_p \oplus M_c + (\{c_k\}_{k=1 \rightarrow K})$ such that: for $x \in \mathcal{D}_U$, $\Pr(M_U(x) \neq M_{\mathcal{D}}(x)) \geq \delta$.

Unlearning on an Entire Class: For classification tasks, another common scenario is to remove an entire class of data which can be regarded as a special case for multiple data points unlearning. Next, we illustrate the application of PRUNE in this setting. Suppose $\mathcal{D}_U = \{x_u, y_{unlearn}\}_{u=1,\dots,r}$ 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 Equation 7. Afterwards, similar to multipoint unlearning, the output of x_u on $M_p \oplus (M_c + c)$ is compared to determine if the erasure effect $y_{unlearn} \rightarrow \hat{y}_{unlearn}$ 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

Algorithm 1 PRUNE-Multipoint

```

1: Input:  $\mathcal{D}_U, M_{\mathcal{D}} = M_p \oplus M_c$ 
2: Output:  $M_U$ 
3: Initialize clusters number  $K$ 
4:  $\mathcal{D}_U^k, \{x_c^k, y_c^k\}_{k=1 \rightarrow K} \leftarrow \text{KMeans}(\mathcal{D}_U, K)$ 
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  $(x_u^k, y_u^k) \in \mathcal{D}_U^k$  do
9:     if  $M_p(x_u^k) \oplus (M_c(x_u^k) + m_k(x_u^k))! = y_u^k$  then
10:       $n_u^k \leftarrow$  according to Eq. 6
11:    else
12:       $\mathcal{D}_{UR} \leftarrow (x_u^k, y_u^k)$ 
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:    $\mathcal{D}_U \leftarrow \mathcal{D}_{UR}$ 
20:   line4-line17
21: until  $1 - \mathcal{D}_{UR}/\mathcal{D}_U > \delta$ 
22:  $M_U = M_{\mathcal{D}}$ 

```

effect not only for \mathcal{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 of CROWN [42], a verification tool for bound propagation, the perturbation of upper boundary $\bar{\Delta}$ and lower boundary $\underline{\Delta}$ that can satisfy $M_{\mathcal{D}}(x_c + \Delta) = y_{unlearn}$ is quickly computed. We use Equation 5 to calculate the activation pattern $S(x_c)$ on $M_{\mathcal{D}}$ 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 Equation 6. The same process is used to generate n'_u for the other data points in \mathcal{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 Equation 8. The overall process of PRUNE for class unlearning is summarized in Algorithm 2.

C. Comparison with Related Techniques

Table I 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

Algorithm 2 PRUNE-Class

```

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

```

TABLE I: Comparison with existing unlearning techniques

	PRUNE	Retrain	SISA [6]	AML [43]	FU [16]
Use of Remaining Data	✗	✓	✓	✓	✗
Use of Erasure Data	✓	✗	✗	✓	✓
Single Point	✓	✓	✓	✓	✓
Training Recording	✗	✗	✓	✓	✗
Certifiability	✓	✗	✗	✓(effect)	✓
Limited Side Effect	✓	✓	✓	✗	✗

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.

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.

V. 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?
- **RQ2:** How does the efficiency of our approach compare with baseline methods?

TABLE II: Details of datasets and models

Dataset	Classes	Features	Training Set	Test Set	Model
Purchase-20	20	600	38758	9689	FC(256)
HAR	6	561	8238	2060	FC(256)
MNIST	10	28×28	60000	10000	FC(256×256)
Fashion-MNIST	10	28×28	60000	10000	FC(256×256)
CIFAR-10	10	32×32×3	50000	10000	VGG16

- **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?

A. Experimental Setup

Datasets and Models. We conduct experiments on five popular classification datasets in machine unlearning research: Purchase-100 [44], Human Activity Recognition (HAR) [45], MNIST [46], Fashion-MNIST [47], and CIFAR-10 [48]. The fully connected (FC) neural network with ReLU activation function is chosen to perform classification prediction on the first four datasets. Details of the dataset and models are in Table II. Considering the application scenario and Property 5 requirement of unlearning, the data points in \mathcal{D}_U are randomly selected from the training set. The parameters for models training are described in [32].

Baselines. We compare our approach with four widely used unlearning mechanisms: 1) Retraining, as the naive unlearning method, trains the model from scratch on $\mathcal{D} \setminus \mathcal{D}_U$. 2) SISA [6] 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) [43] 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 [16] 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.

B. Utility Guarantee

We comprehensively use four metrics to evaluate each unlearning method: the accuracy on the test set A_{tes} , the accuracy on the data not requested to be withdrawn A_{res} , the accuracy on \mathcal{D}_U before and after unlearning $A_{u,b}$, $A_{u,a}$. Note that $A_{u,b}$ and $A_{u,a}$ are intuitive metrics to ease the data holder’s concern by verifying whether the prediction of \mathcal{D}_U changes (forgetting is successful). A_{tes} is used to evaluate the change in model performance before and after the execution of unlearning algorithms. A_{res} evaluates the impact on the rest data. The lower the $A_{u,a}$, the better the unlearning effect. The smaller the change in A_{tes} and A_{res} , the less the unlearning side-effect.

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 III shows the results of the model accuracy on different datasets using different unlearning algorithms. The number of data points in \mathcal{D}_U varies between [100, 200, 300, 500]. Data points in \mathcal{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 remain essentially unchanged on the predictions of \mathcal{D}_U after unlearning, and FU only has a limited decrease. This is because the trained model possesses a certain degree of generalization ability to predict data points that have not been seen before. However, from the perspective of the data holders and auditors, this leaves no intuitive measure of whether or not unlearning has been performed. In contrast, a huge difference (over 90%) emerged between $A_{u,b}$ and $A_{u,a}$ for both AML and PRUNE, meaning both methods are able to confuse the model's judgment on these data points to erase (with $A_{u,a}$ drops to 3% on average for PRUNE). Thus, instead of accessing the model parameters, the auditor can intuitively assess the erasure effectiveness by submitting data to test the model output. Data holders can obtain assured-level of model failure on their individual data.

In terms of overall performance, we observe that AML, despite effectively unlearning data, experiences a significant drop in A_{tes} as the number of forgotten data points increases. 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 submodel. PRUNE minimizes the degradation of the overall performance of the model by making $m(x)$ effective only on \mathcal{D}_U . It makes A_{tes} basically unchanged on the three datasets of Purchase, HAR, and MNIST, and the degradation of A_{tes} on the Fashion-MNIST and CIFAR-10 dataset 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.

C. 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 Figure 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. Except for FU, PRUNE shows comparable efficiency to AML (better than Retrain and SISA) and greater stability. For example, when unlearning 500 samples for MNIST, Fashion-MNIST, and CIFAR-10, the unlearning time spent by AML increased from 0.40 to 0.39 and 0.99, whereas the time taken by PRUNE decreased from 1.05 to 0.79 and 0.69. 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.

D. Privacy Guarantee

We use the membership inference attack proposed in [49] 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. From the results in Table IV, we can observe

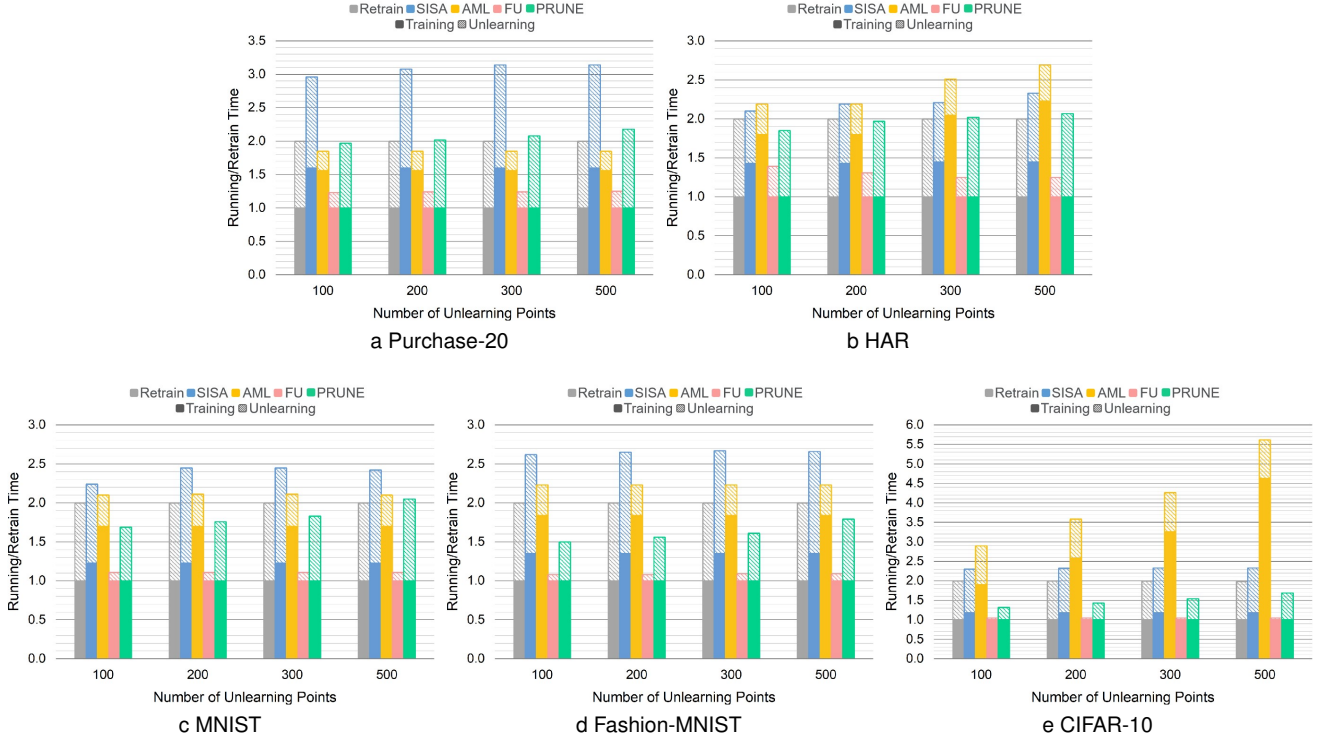


Fig. 3: The efficiency of different methods to unlearn on multiple datasets. The efficiency is measured by summing the normalized running times of the two phases. Lower values represent higher efficiency.

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 the data holders on this basis.

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

E. Class Processing

For unlearning data of the whole category, we consider the performance of the model both on \mathcal{D}_U and on that category data in the test set. Table V shows more detailed statistics obtained by randomly selecting the category labels for unlearning 20 times on the five datasets. A_u and A_{res} 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 unlearning 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 three 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 five 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.

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

F. Hyper-parameter Tuning

Varying the number of centroids K : Before executing our algorithm, it is necessary to determine the number of centroids

TABLE III: Performance comparison when different numbers of data points are unlearned.

Dataset	\mathcal{D}_U	Acc(%)	Retrain	SISA	AML	FU	PRUNE
Purchase	0	<i>Ates</i>	95.08	90.32	95.16	92.76	95.30
		<i>Atra</i>	98.37	91.39	99.44	99.56	98.38
		<i>Ares</i>	95.17	90.27	91.65	80.69	95.30
	100	<i>Ares</i>	98.49	91.41	97.42	81.57	98.13
		<i>A_{u,b}</i>	97.00	91.00	95.00	92.00	99.00
		<i>A_{u,a}</i>	97.00	88.00	0.00	76.00	1.83
		<i>Ates</i>	94.97	90.68	91.22	81.47	95.30
	200	<i>Ares</i>	98.45	91.51	96.88	81.96	97.89
		<i>A_{u,b}</i>	99.50	91.00	100.00	92.00	100.00
		<i>A_{u,a}</i>	95.50	89.50	0.00	77.50	2.30
		<i>Ates</i>	95.17	90.67	90.99	79.24	95.30
	300	<i>Ares</i>	98.47	91.73	98.48	80.16	97.64
		<i>A_{u,b}</i>	98.00	91.33	99.33	91.67	97.67
		<i>A_{u,a}</i>	96.33	89.67	0.00	83.33	2.67
		<i>Ates</i>	95.07	90.10	90.50	82.74	95.30
	500	<i>Ares</i>	98.45	91.32	96.30	82.63	97.15
		<i>A_{u,b}</i>	98.80	91.80	100.00	92.80	98.40
		<i>A_{u,a}</i>	95.20	90.60	0.00	82.80	2.91
HAR	0	<i>Ates</i>	95.05	95.29	94.98	97.78	95.82
		<i>Atra</i>	96.34	95.89	97.74	98.79	97.57
		<i>Ares</i>	95.09	93.87	80.72	97.10	95.82
	100	<i>Ares</i>	96.41	94.63	96.97	98.46	96.47
		<i>A_{u,b}</i>	97.00	97.00	100.00	100.00	100.00
		<i>A_{u,a}</i>	97.00	96.00	0.00	100.00	7.33
		<i>Ates</i>	94.92	93.27	80.23	72.90	95.61
	200	<i>Ares</i>	96.36	94.04	96.38	74.52	96.23
		<i>A_{u,b}</i>	96.00	97.50	99.00	99.50	97.50
		<i>A_{u,a}</i>	95.00	95.00	0.00	81.00	6.33
		<i>Ates</i>	94.90	91.02	80.52	80.94	95.50
	300	<i>Ares</i>	96.15	92.05	96.73	81.23	95.23
		<i>A_{u,b}</i>	97.00	96.67	97.33	98.33	97.67
		<i>A_{u,a}</i>	95.67	91.33	0.00	82.00	5.22
		<i>Ates</i>	95.02	87.99	80.76	73.99	95.11
	500	<i>Ares</i>	96.18	87.77	97.02	73.54	94.07
		<i>A_{u,b}</i>	96.00	96.60	95.20	98.20	98.60
		<i>A_{u,a}</i>	95.80	89.60	0.00	71.20	5.00
MNIST	0	<i>Ates</i>	98.04	96.05	96.37	98.43	98.00
		<i>Atra</i>	99.89	96.14	99.11	99.90	99.64
		<i>Ares</i>	98.17	96.01	86.84	98.58	98.00
	100	<i>Ares</i>	99.89	96.15	97.95	99.94	99.48
		<i>A_{u,b}</i>	100.00	93.00	95.00	100.00	100.00
		<i>A_{u,a}</i>	99.00	92.00	0.00	100.00	3.40
		<i>Ates</i>	98.14	96.10	86.52	93.36	98.00
	200	<i>Ares</i>	99.89	96.16	98.37	92.91	99.32
		<i>A_{u,b}</i>	100.00	98.00	94.00	100.00	99.00
		<i>A_{u,a}</i>	97.50	97.50	0.00	92.00	3.43
		<i>Ates</i>	98.23	96.10	84.15	91.30	98.00
	300	<i>Ares</i>	99.89	96.15	98.91	91.60	99.15
		<i>A_{u,b}</i>	100.00	94.33	97.67	99.67	99.33
		<i>A_{u,a}</i>	98.67	94.00	0.00	92.33	1.33
		<i>Ates</i>	98.06	95.98	83.78	97.71	98.00
	500	<i>Ares</i>	99.89	96.15	96.99	97.46	98.84
		<i>A_{u,b}</i>	100.00	95.20	97.00	99.80	99.20
		<i>A_{u,a}</i>	98.40	94.80	0.00	97.60	2.77
Fashion-MNIST	0	<i>Ates</i>	87.24	86.76	87.11	86.58	87.12
		<i>Atra</i>	89.78	88.72	90.06	89.45	89.55
		<i>Ares</i>	87.03	86.77	78.16	82.88	86.95
	100	<i>Ares</i>	89.62	88.67	90.36	83.21	89.16
		<i>A_{u,b}</i>	87.00	87.00	88.00	90.00	94.00
		<i>A_{u,a}</i>	84.00	86.00	0.00	82.00	3.00
		<i>Ates</i>	87.06	86.65	78.76	83.00	86.67
	200	<i>Ares</i>	89.45	88.62	90.90	83.13	88.78
		<i>A_{u,b}</i>	90.00	89.50	91.00	89.50	89.00
		<i>A_{u,a}</i>	88.50	89.00	0.00	87.50	3.20
		<i>Ates</i>	86.97	86.80	77.76	83.82	86.55
	300	<i>Ares</i>	89.44	88.71	89.28	80.67	88.42
		<i>A_{u,b}</i>	89.00	91.00	87.00	87.66	90.67
		<i>A_{u,a}</i>	88.67	90.00	0.00	80.40	0.80
		<i>Ates</i>	86.80	86.86	75.92	81.48	85.84
	500	<i>Ares</i>	89.27	88.61	90.95	81.49	87.41
		<i>A_{u,b}</i>	90.20	90.80	89.20	87.10	92.40
		<i>A_{u,a}</i>	90.80	90.40	0.00	83.20	0.77
Cifar-10	0	<i>Ates</i>	87.84	86.66	86.73	86.36	87.63
		<i>Atra</i>	97.81	97.29	96.03	99.45	97.46
		<i>Ares</i>	88.13	86.61	75.14	83.72	86.59
	100	<i>Ares</i>	97.94	97.16	90.32	86.08	96.01
		<i>A_{u,b}</i>	96.00	97.00	95.00	93.44	95.00
		<i>A_{u,a}</i>	86.00	97.00	0.00	84.70	2.80
		<i>Ates</i>	86.92	86.65	73.57	83.51	85.33
	200	<i>Ares</i>	96.71	97.17	87.43	85.50	94.65
		<i>A_{u,b}</i>	98.50	96.80	93.38	92.29	94.50
		<i>A_{u,a}</i>	91.00	95.50	0.00	82.60	4.20
		<i>Ates</i>	87.16	86.84	75.64	84.23	84.62
	300	<i>Ares</i>	96.76	97.36	89.37	85.34	93.52
		<i>A_{u,b}</i>	98.67	95.89	95.50	96.26	94.33
		<i>A_{u,a}</i>	89.00	95.57	0.00	83.67	4.50
		<i>Ates</i>	87.51	86.58	74.62	81.12	83.65
	500	<i>Ares</i>	97.41	97.34	90.51	82.52	91.83
		<i>A_{u,b}</i>	97.80	95.40	96.85	94.50	96.60
		<i>A_{u,a}</i>	86.60	94.60	0.00	80.50	3.40

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 \mathcal{D}_U . Furthermore, K also determines the number of optimized $m(x)$ at each iteration, thus influencing the convergence and time complexity of our algorithm. Figure 4 shows the accuracy of the model on \mathcal{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 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 four 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 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. It does not imply that our method lacks competitiveness in terms of efficiency for complex models. As demonstrated by the experiments in Section 5.3, the time cost remains manageable.

Varying the scale of \mathcal{D}_U : The HAR dataset and the MNIST dataset are chosen for discussion here. Figure 5 shows the accuracy of the model on \mathcal{D}_U with the number of iterations when different scales of \mathcal{D}_U are taken. The results of the other three datasets are shown in [32]. 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 \mathcal{D}_U varies. This proves the superiority of our algorithm on multipoint withdrawal. However, as the size of \mathcal{D}_U increases, there is a tendency for PRUNE to converge faster. This trend is most obvious on the HAR dataset. When $|\mathcal{D}_U| = 500$, the average accuracy of the model on \mathcal{D}_U after 3 iterations is 3.1%. It is 1/3 of the accuracy of the model when $|\mathcal{D}_U| = 100$. 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 Figure 5a and 5b). Increasing the size of \mathcal{D}_U alleviates this problem to some extent. A larger \mathcal{D}_U scale implies a broader range of data points to be erased, enhancing the representativeness of our clustering selection before each iteration.

TABLE IV: The success rate (%) of member inference attack on \mathcal{D}_U before and after using different unlearning mechanism when $|\mathcal{D}_U| = 100$. The value indicates the proportion of \mathcal{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		Fashion-MNIST		CIFAR-10	
	before	after	before	after	before	after	before	after	before	after
Retrain	84.30	84.30	84.00	83.70	92.60	83.00	79.40	77.40	84.00	78.00
SISA	84.20	80.00	83.40	79.80	92.60	91.00	79.30	75.00	90.80	87.40
AML	99.00	0.00	89.70	0.00	77.50	0.00	65.80	0.00	73.30	0.00
FU	85.30	70.50	84.80	84.70	90.80	88.90	83.80	78.30	91.00	90.20
PRUNE	81.70	0.00	81.40	4.04	92.90	1.01	79.80	0.00	84.80	5.00

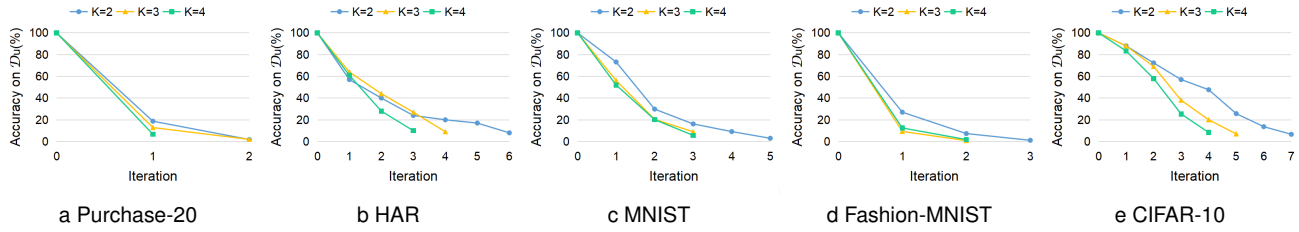


Fig. 4: The convergence of PRUNE w.r.t the number of centroids K ($|\mathcal{D}_U| = 100$).

TABLE V: Results of the entire class unlearning on the datasets using PRUNE.

Acc(%)	Phase	Purchase-20	HAR	MNIST	Fashion-MNIST	CIFAR-10
A_u	before	98.14	97.51	98.95	93.28	95.52
	after	0.00	0.00	0.13	0.00	0.00
A_{tes_u}	before	94.75	95.91	97.4	89.37	87.70
	after	7.15	1.55	17.08	12.16	9.02
A_{res}	before	98.40	97.61	99.80	88.80	97.72
	after	90.68	95.88	99.26	85.86	91.14
A_{tes_r}	before	95.31	95.80	98.08	87.63	87.95
	after	91.24	90.01	96.86	82.83	82.99

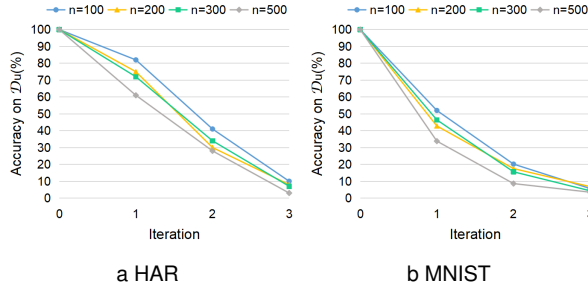


Fig. 5: The convergence of PRUNE w.r.t the scale of \mathcal{D}_U ($K = 4$).

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

VI. 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 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.

VII. RELATED WORK

A. Machine Unlearning

Considering the privacy issues involved in machine learning, Cao and Yang [50] 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 [5], [20], [51]. For instance, Guo et al [4] incorporated the idea of differential privacy [52], [53] 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 [6] 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 [11] and graph neural networks [10]. 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 [43] saved gradient information of each training batch and quickly removed data by subtracting affected gradients. Thudi et al [17] 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 [21], 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.

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 [54]. 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 [23] 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 [26] 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 [27] 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.

B. 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 [55] 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 [56] 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/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 [36] 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 [31] 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 [28] 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 [57] 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 [39] 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.

VIII. 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 [32].

REFERENCES

- [1] G. D. P. Regulation, "General data protection regulation (gdpr)," *Intersoft Consulting*, Accessed in October, vol. 24, no. 1, 2018.
- [2] A. A. Ginart, M. Y. Guan, G. Valiant, and J. Zou, "Making ai forget you: data deletion in machine learning," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 2019, pp. 3518–3531.
- [3] S. Shintre, K. A. Roundy, and J. Dhaliwal, "Making machine learning forget," in *Privacy Technologies and Policy: 7th Annual Privacy Forum, APF 2019, Rome, Italy, June 13–14, 2019, Proceedings 7*. Springer, 2019, pp. 72–83.
- [4] C. Guo, T. Goldstein, A. Hannun, and L. Van Der Maaten, "Certified data removal from machine learning models," in *International Conference on Machine Learning*. PMLR, 2020, pp. 3832–3842.
- [5] Q. P. Nguyen, B. Kian, H. Low, and P. Jaillet, "Variational bayesian unlearning," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020, pp. 16 025–16 036.
- [6] L. Bourtole, V. Chandrasekaran, C. A. Choquette-Choo, H. Jia, A. Travers, B. Zhang, D. Lie, and N. Papernot, "Machine unlearning," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 141–159.
- [7] V. Gupta, C. Jung, S. Neel, A. Roth, S. Sharifi-Malvajerdi, and C. Waites, "Adaptive machine unlearning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 16 319–16 330, 2021.
- [8] T. Shibata, G. Irie, D. Ikami, and Y. Mitsuzumi, "Learning with selective forgetting," in *30th International Joint Conference on Artificial Intelligence, IJCAI 2021*. International Joint Conferences on Artificial Intelligence, 2021, pp. 989–996.
- [9] H. Yan, X. Li, Z. Guo, H. Li, F. Li, and X. Lin, "Arcane: An efficient architecture for exact machine unlearning," in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, 2022, pp. 4006–4013.
- [10] M. Chen, Z. Zhang, T. Wang, M. Backes, M. Humbert, and Y. Zhang, "Graph unlearning," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 499–513.
- [11] C. Chen, F. Sun, M. Zhang, and B. Ding, "Recommendation unlearning," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 2768–2777.
- [12] S. Schelter, S. Grafberger, and T. Dunning, "Hedgecut: Maintaining randomised trees for low-latency machine unlearning," in *Proceedings of the 2021 International Conference on Management of Data*, 2021, pp. 1545–1557.
- [13] Y. Li, C.-H. Wang, and G. Cheng, "Online forgetting process for linear regression models," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021, pp. 217–225.
- [14] Y. Liu, L. Xu, X. Yuan, C. Wang, and B. Li, "The right to be forgotten in federated learning: An efficient realization with rapid retraining," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 2022, pp. 1749–1758.
- [15] Z. Zhang, Y. Zhou, X. Zhao, T. Che, and L. Lyu, "Prompt certified machine unlearning with randomized gradient smoothing and quantization," *Advances in Neural Information Processing Systems*, vol. 35, pp. 13 433–13 455, 2022.
- [16] A. Warnecke, L. Pirch, C. Wressnegger, and K. Rieck, "Machine unlearning of features and labels," in *Proc. of the 30th Network and Distributed System Security (NDSS)*, 2023.
- [17] A. Thudi, G. Deza, V. Chandrasekaran, and N. Papernot, "Unrolling sgd: Understanding factors influencing machine unlearning," in *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2022, pp. 303–319.
- [18] E. Chien, C. Pan, and O. Milenkovic, "Efficient model updates for approximate unlearning of graph-structured data," in *International Conference on Learning Representations*, 2023.
- [19] A. Golatkar, A. Achille, and S. Soatto, "Eternal sunshine of the spotless net: Selective forgetting in deep networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9304–9312.
- [20] Z. Izzo, M. A. Smart, K. Chaudhuri, and J. Zou, "Approximate data deletion from machine learning models," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021, pp. 2008–2016.
- [21] G. Wu, M. Hashemi, and C. Srinivasa, "Puma: Performance unchanged model augmentation for training data removal," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 8, 2022, pp. 8675–8682.
- [22] J. Wu, Y. Yang, Y. Qian, Y. Sui, X. Wang, and X. He, "Gif: A general graph unlearning strategy via influence function," in *Proceedings of the ACM Web Conference 2023*, 2023, pp. 651–661.
- [23] A. Thudi, H. Jia, I. Shumailov, and N. Papernot, "On the necessity of auditable algorithmic definitions for machine unlearning," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 4007–4022.
- [24] J. Gao, S. Garg, M. Mahmood, and P. N. Vasudevan, "Deletion inference, reconstruction, and compliance in machine (un) learning," *Proceedings on Privacy Enhancing Technologies*, vol. 3, pp. 415–436, 2022.
- [25] J. Zhou, H. Li, X. Liao, B. Zhang, W. He, Z. Li, L. Zhou, and X. Gao, "Audit to forget: A unified method to revoke patients' private data in intelligent healthcare," *bioRxiv*, pp. 2023–02, 2023.
- [26] A. K. Tarun, V. S. Chundawat, M. Mandal, and M. Kankanalli, "Fast yet effective machine unlearning," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.
- [27] M. Chen, W. Gao, G. Liu, K. Peng, and C. Wang, "Boundary unlearning," *arXiv preprint arXiv:2303.11570*, 2023.
- [28] B. Goldberger, G. Katz, Y. Adi, and J. Keshet, "Minimal modifications of deep neural networks using verification," *EPIC Series in Computing*, vol. 73, pp. 260–278, 2020.
- [29] G. Dong, J. Sun, X. Wang, X. Wang, and T. Dai, "Towards repairing neural networks correctly," in *2021 IEEE 21st International Conference on Software Quality, Reliability and Security (QRS)*. IEEE, 2021, pp. 714–725.
- [30] M. Sotoudeh and A. V. Thakur, "Provable repair of deep neural networks," in *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, 2021, pp. 588–603.
- [31] J. Sohn, S. Kang, and S. Yoo, "Arachne: Search based repair of deep neural networks," *ACM Transactions on Software Engineering and Methodology*, 2022.
- [32] Anonymous, "The repository of code and data to support patching based repair framework for certifiable unlearning," <https://github.com/dummyPRUNE2024>, 2023.
- [33] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," *arXiv preprint arXiv:1708.06733*, 2017.
- [34] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *2nd International Conference on Learning Representations, ICLR 2014*, 2014.
- [35] W. Samek, T. Wiegand, and K.-R. Müller, "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*, vol. 1, pp. 1–10, 2017.
- [36] M. Usman, D. Gopinath, Y. Sun, Y. Noller, and C. S. Păsăreanu, "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*. Springer, 2021, pp. 3–25.
- [37] B. Sun, J. Sun, L. H. Pham, and J. Shi, "Causality-based neural network repair," in *Proceedings of the 44th International Conference on Software Engineering*, 2022, pp. 338–349.
- [38] M. Sotoudeh and A. V. Thakur, "Correcting deep neural networks with small, generalizing patches," in *NeurIPS 2019 Workshop on Safety and Robustness in Decision Making*, 2019.

- [39] F. Fu and W. Li, "Sound and complete neural network repair with minimality and locality guarantees," in *International Conference on Learning Representations*, 2022.
- [40] G.-H. Lee, D. Alvarez-Melis, and T. S. Jaakkola, "Towards robust, locally linear deep networks," in *International Conference on Learning Representations*, 2019.
- [41] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski, *Robust optimization*. Princeton university press, 2009, vol. 28.
- [42] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel, "Efficient neural network robustness certification with general activation functions," *Advances in neural information processing systems*, vol. 31, 2018.
- [43] L. Graves, V. Nagisetty, and V. Ganesh, "Amnesiac machine learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 13, 2021, pp. 11 516–11 524.
- [44] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE symposium on security and privacy (SP)*. IEEE, 2017, pp. 3–18.
- [45] E. Bulbul, A. Cetin, and I. A. Dogru, "Human activity recognition using smartphones," in *2018 2nd international symposium on multidisciplinary studies and innovative technologies (ismsit)*. IEEE, 2018, pp. 1–6.
- [46] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [47] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.
- [48] A. Krizhevsky *et al.*, "Learning multiple layers of features from tiny images," <https://www.cs.toronto.edu/kriz/learning-features-2009-TR.pdf>, 2009.
- [49] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," in *2018 IEEE 31st computer security foundations symposium (CSF)*. IEEE, 2018, pp. 268–282.
- [50] Y. Cao and J. Yang, "Towards making systems forget with machine unlearning," in *2015 IEEE symposium on security and privacy*. IEEE, 2015, pp. 463–480.
- [51] S. Neel, A. Roth, and S. Sharifi-Malvajerdi, "Descent-to-delete: Gradient-based methods for machine unlearning," in *Algorithmic Learning Theory*. PMLR, 2021, pp. 931–962.
- [52] C. Dwork, "Differential privacy," in *Automata, Languages and Programming: 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II* 33. Springer, 2006, pp. 1–12.
- [53] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 308–318.
- [54] Z. Shumaylov, D. Kazhdan, Y. Zhao, N. Papernot, M. A. Erdogdu, and R. J. Anderson, "Manipulating sgd with data ordering attacks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 18 021–18 032, 2021.
- [55] X. Ren, B. Yu, H. Qi, F. Juefei-Xu, Z. Li, W. Xue, L. Ma, and J. Zhao, "Few-shot guided mix for dnn repairing," in *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2020, pp. 717–721.
- [56] S. Ma, Y. Liu, W.-C. Lee, X. Zhang, and A. Grama, "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*, 2018, pp. 175–186.
- [57] T. S. Borkar and L. J. Karam, "Deepcorrect: Correcting dnn models against image distortions," *IEEE Transactions on Image Processing*, vol. 28, no. 12, pp. 6022–6034, 2019.



Xuran Li received a B.S. degree in intelligent manufacturing engineering from Tongji University in 2022. She is currently pursuing the Ph.D. degree in Cyberspace Security at Zhejiang University. Her research interests include data privacy and artificial intelligence security.



during 2018–2019. His research interests include formal methods, software engineering, cyber security, and machine learning.

Jingyi Wang received the bachelor's degree in information engineering from Xi'an Jiaotong University, in 2013, and the Ph.D. degree from Singapore University of Technology and Design, in 2018. He is currently a Tenure-Track Assistant Professor with the College of Control Science and Engineering, Zhejiang University, China. He was a Research Fellow with the School of Computing, National University of Singapore, during 2019–2020 and with Information Systems Technology and Design Pillar, Singapore University of Technology and Design, during 2018–2019. His research interests include formal methods, software engineering, cyber security, and machine learning.



Xiaohan Yuan received a B.S. degree in intelligent manufacturing engineering from Tongji University, Shanghai, China, in 2023. He is now pursuing a M.S. degree in the College of Control Science and Engineering, Zhejiang University, Zhejiang, China. His research interests include privacy protection and AI security.



Peixin Zhang is currently a research scientist at the School of Computing and Information Systems, Singapore Management University. He received his bachelor's and Ph.D. degrees in computing science from Zhejiang University in 2016 and 2022, respectively. He was a visiting student at Singapore University of Technology and Design in 2017, and Singapore Management University from 2019 to 2020, respectively. His research interests include software engineering and artificial intelligence, especially software engineering for artificial intelligence.



Zhan Qin (Senior Member, IEEE) is currently a ZJU100 Young Professor, with both the College of Computer Science and Technology and the Institute of Cyberspace Research (ICSR) at Zhejiang University, China. He was an assistant professor at the Department of Electrical and Computer Engineering in the University of Texas at San Antonio after receiving the Ph.D. degree from the Computer Science and Engineering department at State University of New York at Buffalo in 2017. His current research interests include data security and privacy, secure computation outsourcing, artificial intelligence security, and cyber-physical security in the context of the Internet of Things. His works explore and develop novel security sensitive algorithms and protocols for computation and communication on the general context of Cloud and Internet devices.



Zhibo Wang (Senior Member, IEEE) received the B.E. degree in Automation from Zhejiang University, China, in 2007, and his Ph.D degree in Electrical Engineering and Computer Science from University of Tennessee, Knoxville, in 2014. He is currently a Professor with the School of Cyber Science and Technology, Zhejiang University, China. His currently research interests include Internet of Things, AI security, data security and privacy. He is a Senior Member of IEEE and a Member of ACM.



Kui Ren (Fellow, IEEE) received the PhD degree in electrical and computer engineering from Worcester Polytechnic Institute. He is the Dean and Professor of the College of Computer Science and Technology, at Zhejiang University, where he also directs the Institute of Cyber Science and Technology. Before that, he was SUNY Empire Innovation professor at the State University of New York at Buffalo, USA. His current research interests include data security, IoT security, AI security, and privacy. He received many recognitions including Guohua Distinguished

Scholar Award of ZJU, IEEE CISTC Technical Recognition Award, SUNY Chancellor's Research Excellence Award, Sigma Xi Research Excellence Award, NSF CAREER Award, etc. He has published extensively in peer-reviewed journals and conferences and received the Test-of-time Paper Award from IEEE INFOCOM and many Best Paper Awards, including ACM MobiSys, IEEE ICDCS, IEEE ICNP, IEEE Globecom, ACM/IEEE IWQoS, etc. His h-index is 93, with a total citation exceeding 48,000 according to Google Scholar. He is a fellow of the ACM. He is a frequent reviewer for funding agencies internationally and serves on the editorial boards of many IEEE and ACM journals. Among others, he currently serves as chair of SIGSAC of ACM China Council, a member of ACM ASIACCS steering committee, and a member of S&T Committee of Ministry of Education of China.