

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

Xuran Li^{1,3}, Jingyi Wang^{1,3*}, Xiaohan Yuan¹, Peixin Zhang²

¹ Zhejiang University, Hangzhou, China ² Singapore Management University, Singapore

³ Huzhou Institute of Industrial Control Technology, Huzhou, China

{xuranli1005, wangjyee, xiaohanyuan}@zju.edu.cn, pxzhang94@gmail.com

Abstract—It is often desirable to remove (a.k.a. unlearn) specific training data from a neural network model, especially to uphold the data holder’s *right to be forgotten* as mandated by recent regulations. Beyond privacy concerns, unlearning is also increasingly relevant in industrial settings, such as in adaptive control systems or predictive maintenance pipelines, where removing outdated or faulty sensor data is critical for system safety and performance. Existing unlearning methods involve training alternative models with remaining data, which may be costly. 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. Extensive experiments on multiple categorical datasets demonstrates our approach’s effectiveness, achieving measurable unlearning while preserving the model’s performance.

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

I. INTRODUCTION

In the era of data-driven machine learning, legal frameworks such as the GDPR [1] grant users the *right to be forgotten*, allowing them to request removal of their data from trained models [2]. To support the request of data erasure (or removal), the study on *machine unlearning* [3]–[6] 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 Arguably, the most intuitive way to unlearn is to retrain an alternative model M_r after removing the data to be withdrawn from the training set. 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. 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 [3], [7]. They often perform additional operations during the model training phase to reduce the cost of retraining by, for example, either slicing the data [3] or segmenting the training [5]. Approximate unlearning [2], [6], [8], [9], 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 [10], [11].

In this work, we propose a **Patching based Repair** framework for certifiable **UN**learning (**PRUNE**). It draws an analogy between data removal in unlearning and error correction in *neural network repair* [12]–[14]. 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. This approach is particularly relevant for scenarios where real-world systems, such as industrial automation or adaptive control pipelines. They must quickly forget outdated or incorrect sensor records without halting operation or undergoing costly retraining. 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 propose a new approach that bridges machine unlearning and neural network repair. By applying a carefully designed patch network, we directly falsify the model’s prediction on the withdrawn data.
- To preserve the model’s performance on the remaining data, we ensure that the patch network is both minimal and localized.
- We evaluate our approach on multiple categorical datasets [15], showing effective forgetting with competitive efficiency and resource usage.

II. METHODOLOGY

A. Problem Definition

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.

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 $x_u \in \mathcal{D}_U$,

$$Obj_1^u : \exists l \neq y_u, M_U^l(x_u) - M_U^{y_u}(x_u) > 0 \quad (1)$$

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

Besides the unlearning objective in Equation (1), similar to the repair task, we also have to pay attention to the overall performance of the model on the remaining data. The ideal situation is that the model produces no change in label judgments on $\mathcal{D}_R$. Thus, the other objective in Equation (2) is to make the performance change on the remaining data as small as possible. It is formulated as

$$Obj_2^u : \min \frac{1}{|\mathcal{D}_R|} \sum_{i=1}^{|\mathcal{D}_R|} |\arg \max_{l \in \mathcal{Y}} M_U^l(x_i) - \arg \max_{l \in \mathcal{Y}} M_{\mathcal{D}}^l(x_i)| \quad (2)$$

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.

B. The PRUNE Framework

In the following, we present our PRUNE in details. As shown in Figure 1, the key idea of PRUNE is to generate a targeted "patch" network on the original model $M_{\mathcal{D}}$ 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 [15].

Theorem 1: Given a neural network model $M_{\mathcal{D}} = M_p \oplus M_c$ 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) $M_U(x_u) \neq M_{\mathcal{D}}(x_u)$; and 2) for $x \in \mathcal{D} / x_u$, $M_U(x) = M_{\mathcal{D}}(x)$.

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 1, 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 [16].

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

$$S_j^i(x_u) = \begin{cases} (\nabla_{x_u} z_j^i)^T \bar{x}_u + z_j^i - (\nabla_{x_u} z_j^i)^T x_u \geq 0, z_j^i \geq 0 \\ (\nabla_{x_u} z_j^i)^T \bar{x}_u + z_j^i - (\nabla_{x_u} z_j^i)^T x_u \leq 0, z_j^i < 0 \end{cases} \quad (3)$$

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 2 (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 [17].

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

$$n_S(x_u, \lambda) = \text{ReLU} \left(\sum_i n(b_i - a_i x_u, \lambda) - N + 1 \right) \quad (4)$$

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 $M_{\mathcal{D}} = M_p \oplus M_c$, 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

$$\begin{cases} \min_{C,d} \max_{x \in S(x_u)} |m(x)| \\ M_U(x) = M_p(x) \oplus (M_c(x) + m(x)) \\ \forall x \in S(x_u), M_U^{y_u}(x) - M_U^l(x) > 0, l \neq y_u \in \mathcal{Y} \end{cases} \quad (5)$$

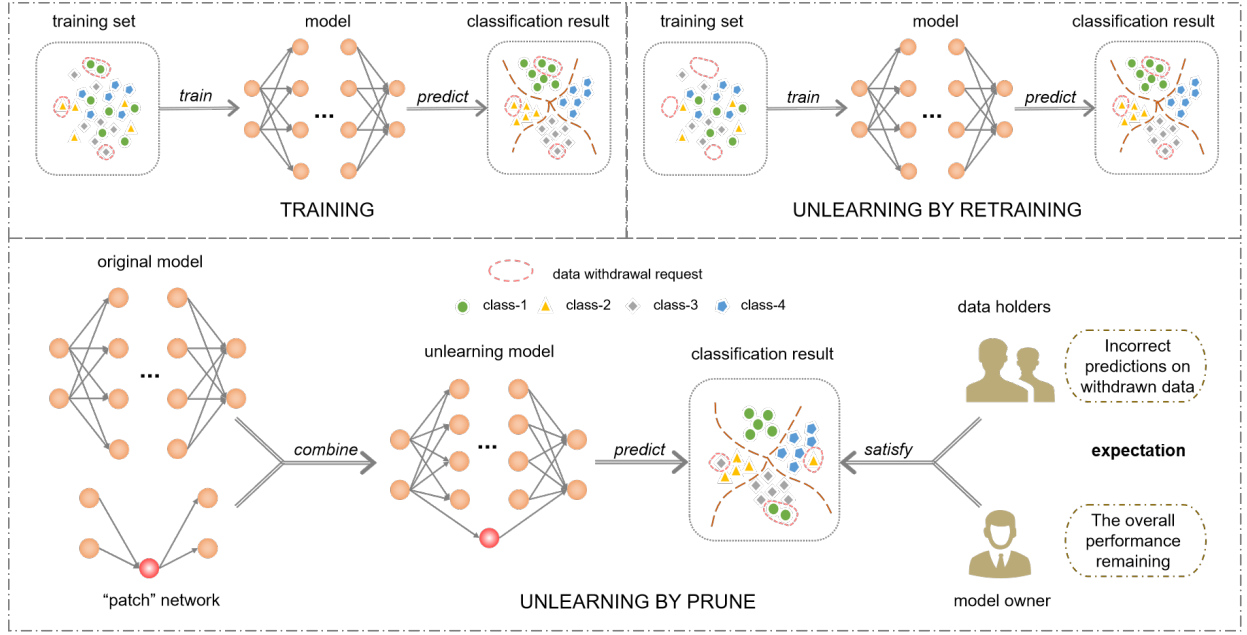


Fig. 1. The framework of PRUNE.

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

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

$$c_S(x_u) = \text{ReLU}(m_S(x_u) + H \cdot n_S(x_u, \lambda) - H) - \text{ReLU}(-m_S(x_u) + H \cdot n_S(x_u, \lambda) - H) \quad (6)$$

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. Next, we further illustrate how to efficiently apply PRUNE to handle unlearning of multiple data points.

Unlearning on Multiple Data Points: Note that the main cost of our algorithm is on the optimization of Equation 5. 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 $x_{u=1, \dots, r'}$ are grouped into K clusters using K-Means, where each cluster has a centroid x_c^k . 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 5. 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 4. 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

$$c_k(x, \lambda) = \text{ReLU}\left(m_k(x) + \max_{1 \leq u \leq r'} \{n_u^k(x, \lambda)\} \cdot H_u - H_u\right) - \text{ReLU}\left(-m_k(x) + \max_{1 \leq u \leq r'} \{n_u^k(x, \lambda)\} \cdot H_u - H_u\right) \quad (7)$$

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 $x \in \mathcal{D}_U$, $\Pr(M_U(x) \neq M_{\mathcal{D}}(x)) \geq \delta$.

III. EXPERIMENTS

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

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. 5
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. 4
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. 7
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:  $M_U = M_{\mathcal{D}}$ 

```

TABLE I
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)
CIFAR-10	10	32×32×3	50000	10000	VGG16

- **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?
- **RQ3:** How our approach is affected by hyperparameters?

A. Experimental Setup

Datasets and Models. We conduct experiments on four popular classification datasets in machine unlearning research: Purchase-100 [18], Human Activity Recognition (HAR) [19], MNIST [20], and CIFAR-10 [21]. Details of the dataset and models are in Table I. Considering the application scenario, the data points in $\mathcal{D}_U$ are randomly selected from the training set. The confusion labels are randomly selected. The parameters for models training are described in [15].

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 [3] shards the training data and then trains them separately. It yields predictions obtained by submodel majority voting. 3) AML [22] 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. 4) FU [23] performs closed updates of model parameters based on

influence functions to forget specific training data labels or features. Considering the impact on model performance, we use the second-order update method for FU.

B. 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_{tes} , 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 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. In the case of single-point unlearning, PRUNE can achieve complete forgetting, as theoretically guaranteed in Theorem 1.

To evaluate unlearning multiple data points, Table II reports model accuracy under different unlearning algorithms, with $\mathcal{D}_U$ sizes ranging from 100 to 500, sampled randomly from training batches. For methods like retraining and SISA, predictions on $\mathcal{D}_U$ barely change after unlearning, and FU shows only a slight drop—reflecting the model’s generalization ability. However, this offers little assurance to data holders or auditors that unlearning has occurred. In contrast, both AML and PRUNE yield a significant drop (over 90%) in accuracy on $\mathcal{D}_U$, effectively “confusing” the model on forgotten data. This allows auditors to verify unlearning via model outputs, without inspecting parameters. Notably, AML incurs considerable performance loss on the remaining data. For example, it suffers from 12.59% drops on the accuracy of the test set for MNIST after unlearning 500 samples. PRUNE, however, confines changes to $\mathcal{D}_U$, keeping A_{tes} nearly unchanged on Purchase, HAR, and MNIST, and within 4% on CIFAR-10, demonstrating strong performance preservation.

C. Efficiency Comparison

We evaluate the efficiency of different unlearning methods by recording the total time for training and unlearning, normalized by the training-from-scratch time (i.e., Retrain). As shown in Figure 2, PRUNE and FU match Retrain in training time, while AML and SISA are significantly slower, especially AML, whose cost increases with more unlearning data due to batch-wise parameter updates. AML also requires training-time instrumentation and may introduce privacy risks. In unlearning time, FU is the fastest owing to its closed-form update. PRUNE performs comparably to AML and is more stable, particularly on complex models like VGG16, where it only modifies the fully connected layers. Overall, PRUNE offers a strong balance between efficiency and applicability to real-world, large-scale models.

D. Hyper-parameter Tuning

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

TABLE II
PERFORMANCE COMPARISON WHEN DIFFERENT NUMBERS OF DATA POINTS ARE UNLEARNED.

Dataset	$\mathcal{D}_U$	Acc(%)	Retrain	SISA	AML	FU	PRUNE	Dataset	$\mathcal{D}_U$	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		100	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		200	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		300	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		500	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		100	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		200	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		300	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		500	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

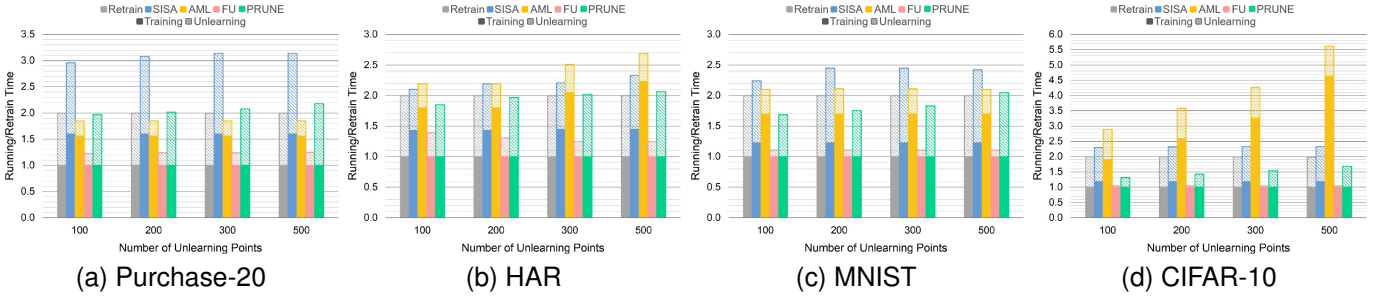


Fig. 2. The efficiency of different methods to unlearn on multiple datasets.

K . That is, the number of representative data points chosen by clustering. K impacts both the quality of the generated patch network and the number of optimized $m(x)$ per iteration, thus affecting convergence and time complexity. As shown in Figure 3, increasing K reduces the required number of iterations, but raises per-iteration computational cost. Ignoring other factors, the overall time complexity is approximately $\mathcal{O}(K \times \text{Iteration})$. For image datasets, we find that $K = 2$ offers the lowest cost, as fewer centroids can represent larger clusters and influence more data points through each $m(x)$.

Varying the scale of $\mathcal{D}_U$: We evaluate convergence behavior on the HAR and MNIST datasets, with results shown in Figure 4. Other datasets' in [15]. Despite varying iteration counts, all settings yield consistent conclusions. 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 $|\mathcal{D}_U| = 100$.

IV. 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. In order to support further development of certifiable machine unlearning, we make all the codes available at the public repository [15].

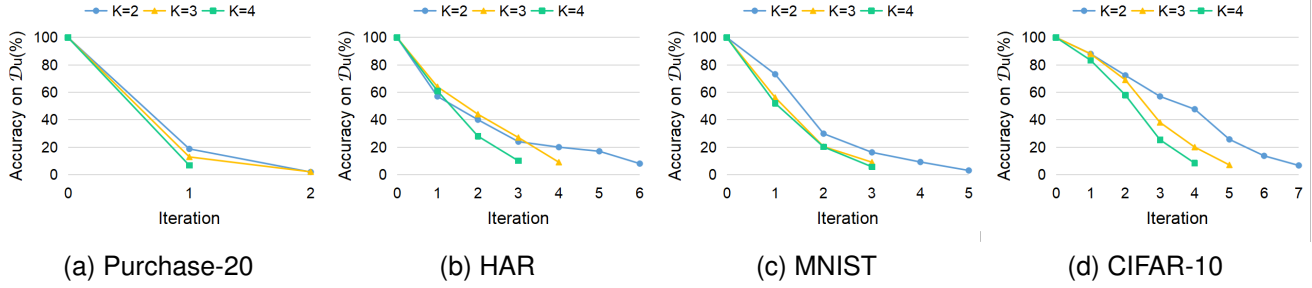


Fig. 3. The convergence of PRUNE w.r.t the number of centroids K ($|\mathcal{D}_U| = 100$).

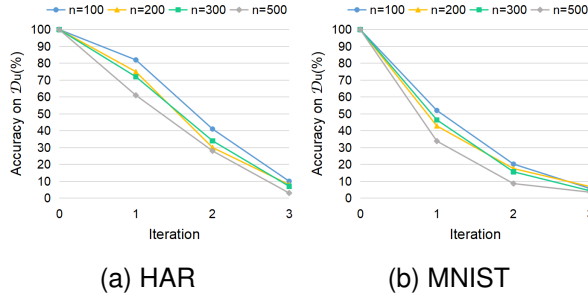


Fig. 4. The convergence of PRUNE w.r.t the scale of $\mathcal{D}_U$ ($K = 4$).

ACKNOWLEDGMENT

This research is supported by the State Key Laboratory of Industrial Control Technology, China (Grant No. ICT2024C01).

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] 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.
- [4] Y. Zhou, D. Zheng, Q. Mo, R. Lu, K.-Y. Lin, and W.-S. Zheng, “Decoupled distillation to erase: A general unlearning method for any class-centric tasks,” in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 20 350–20 359.
- [5] 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.
- [6] J. Foster, S. Schoepf, and A. Brintrup, “Fast machine unlearning without retraining through selective synaptic dampening,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 38, no. 11, 2024, pp. 12 043–12 051.
- [7] X. Hu, D. Li, B. Hu, Z. Zheng, Z. Liu, and M. Zhang, “Separate the wheat from the chaff: Model deficiency unlearning via parameter-efficient module operation,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 16, 2024, pp. 18 252–18 260.
- [8] 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.
- [9] E. Chien, C. Pan, and O. Milenkovic, “Efficient model updates for approximate unlearning of graph-structured data,” in *International Conference on Learning Representations*, 2023.
- [10] 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.
- [11] 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.
- [12] 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.
- [13] J. Ma, P. Yang, J. Wang, Y. Sun, C.-C. Huang, and Z. Wang, “Vere: Verification guided synthesis for repairing deep neural networks,” in *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*, 2024, pp. 1–13.
- [14] J. Sohn, S. Kang, and S. Yoo, “Arachne: Search based repair of deep neural networks,” *ACM Transactions on Software Engineering and Methodology*, 2022.
- [15] Anonymous, “The repository of code and data to support patching based repair framework for certifiable unlearning,” <https://github.com/dummyPRUNE2024>, 2024.
- [16] G.-H. Lee, D. Alvarez-Melis, and T. S. Jaakkola, “Towards robust, locally linear deep networks,” in *International Conference on Learning Representations*, 2019.
- [17] F. Fu and W. Li, “Sound and complete neural network repair with minimality and locality guarantees,” in *International Conference on Learning Representations*, 2022.
- [18] 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.
- [19] 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.
- [20] 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.
- [21] A. Krizhevsky *et al.*, “Learning multiple layers of features from tiny images,” <https://www.cs.toronto.edu/kriz/learning-features-2009-TR.pdf>, 2009.
- [22] 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.
- [23] 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.