Machine Learning (2025) 114:247
https://doi.org/10.1007/s10994-025-06897-9

The forget-set identification problem

Andrea D'Angelo[1] · **Francesco Gullo**[1] · **Giovanni Stilo**[2,3]

Received: 24 April 2025 / Revised: 25 August 2025 / Accepted: 17 September 2025

Abstract

Machine Unlearning (MU) is the problem of removing the influence of user's unwanted evidence from a trained machine-learning model. MU is typically formulated so that the input unwanted evidence corresponds to a subset of the training set utilized to train the model upstream, which is commonly referred to as the "forget set". However, this requirement is often difficult to satisfy in real-world scenarios, as users may be unaware of the peculiarities of the training set or simply they do not have access to it. In a more realistic setting, users provide their unwanted evidence in a form that is more abstract than or anyway different from a precise subset of training data. In such cases, executing MU methods requires an essential and challenging preliminary step, which, to the best of our knowledge, has never been addressed so far: identifying the forget set based on user's unwanted evidence. In this paper, we fill this important gap in the MU literature and introduce the Forget-Set Identification (ForSId) problem: given a trained machine-learning model, an "unwanted set" of samples (evidence to unlearn), and a "wanted set" of samples (evidence to retain), identify the forget set as a subset of the training set, such that the similarity in the predictions of the original model and the model retrained on the training data remaining after the removal of the forget set is: *(i)* low on the unwanted set, indicating that the unwanted samples have been effectively unlearned by the model, and *(ii)* high on the wanted set, to ensure that the model keeps its original performance on the data to be retained. We define ForSId as an optimization problem, prove its NP-hardness, and devise an algorithm based on a theoretical connection to Red-Blue Set Cover. Our ForSId is a novel complementary problem to MU. It serves as a foundational step to be performed before executing MU methods, allowing for extending the range of applicability of MU to all those settings where user's unlearning evidence does not correspond to (or is too hard to be directly expressed in terms of) a forget set. We conduct extensive experiments based on the exact unlearning task (which is the most reliable one) on several real-world datasets and settings, involving nontrivial baselines. Results demonstrate high performance of our proposed algorithm and clear superiority over the baselines.

Editors: Riccardo Guidotti, Anna Monreale, Dino Pedreschi.

Extended author information available on the last page of the article

Published online: 15 October 2025

Keywords Forget set · Machine Unlearning · Trustworthy AI · Model Editing · Red-Blue set cover · Artificial Intelligence · Neural Network

1 Introduction

Machine Unlearning (MU) (Cao & Yang, 2015) (Fig. 1) is the process of removing the influence of specific training samples from a trained machine-learning model.

In principle, this can be done by a complete retraining of the original model on the training set resulting from the removal of the unwanted training samples (Xu et al., 2024). However, a major challenge in MU is to accomplish the unlearning avoiding to retrain models from scratch, as performing this for every individual request is computationally intensive, environmentally unsustainable, and economically impractical (Ginart et al., 2019).

MU has become an effective solution to address increasing demands for data privacy and regulatory compliance, particularly in the context of the "Right to be Forgotten" (Dang, 2021; Kwak et al., 2017), which grants individuals the request to remove their personal information from the systems of any service provider (Kwak et al., 2017). Attention to this context has been further fostered by regulations such as the AI Act European (2021) and the General Data Protection Regulation (GDPR) (Mantelero, 2013).

However, the use of MU goes well beyond privacy contexts. In fact, machine-learning systems are increasingly trained on massive datasets scraped from the Internet or collected from diverse sources. These datasets often contain malicious, harmful, or "not safe for work" content as well as inherent biases (Birhane et al., 2021; Gandikota et al., 2023). These biases can lead to unfair treatment of certain groups of individuals or to the propagation of discriminatory behavior within models, raising significant ethical concerns (Yu et al., 2023; Chen et al., 2023). Furthermore, including such a content in machine-learning models poses legal risks, particularly when these models are deployed in real-world applications.

1.1 Motivation

Machine Unlearning methods are typically formulated so that the subset of training samples to be unlearned, commonly referred to as the "*forget set*", is provided as input. The main goal in MU is to come up with a new model which is as much equivalent as possible to the model that would be obtained by retraining the original one on the training set resulting from the removal of the forget set from it. This way, the resulting model is also expected to keep its performance on what is called the "*retain set*", which typically corresponds to the difference of the training set and the forget set. All of this needs to be ideally carried out without retraining the original model from scratch (Xu et al., 2024).

However, the requirement of having the forget set as input is so restrictive that it might prevent MU from being profitably applied in many real-world scenarios. For instance, individuals requesting data deletion may lack sufficient technical expertise to define the forget set by themselves (Wachter et al., 2017). Also, it is highly likely that they have no access to the training data, as illustrated in Fig. 1. In some other cases, the evidence to be unlearned is simply too complex to be directly expressed as a forget set (see applications below). In a more realistic setting, users know what needs to be unlearned in a more general or abstract sense, rather than as a specific subset of training samples. In cases like these, in order to

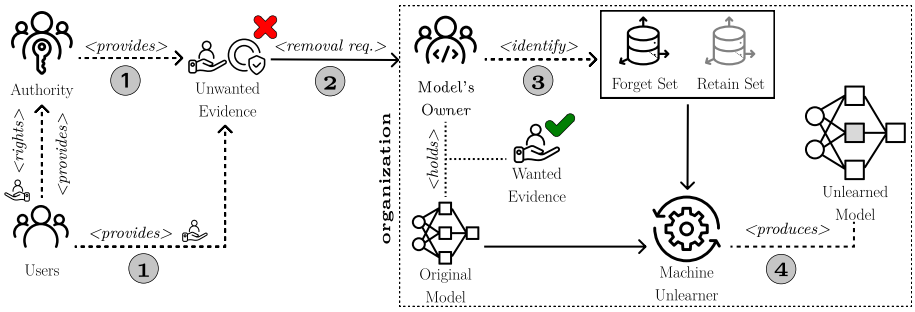

Fig. 1 Overview of the overall Machine Unlearning (MU) process. Directly or indirectly (e.g., passing through some authority), users exercise their right to be forgotten (Step 1). To this end, in Step 2, they provide unwanted evidence that is required to be unlearned through a removal request to machine-learning model's owner. Technically, in Step 3, the organization (that is supposed to hold the wanted evidence to be maintained too) identifies the forget set, as a subset of the upstream training set, and, optionally, a retain set. In Step 4, a MU technique is applied to the original model so as to produce the desired one where user's unwanted evidence has been unlearned. So far, in the MU landscape, Steps 1–3 have been treated as a unique one, i.e., it has been assumed that users provide their unwanted evidence directly in the form of a forget set. This is unrealistic in many scenarios. The Forget-Set Identification (ForSId) problem we introduce in this work is aimed at overcoming precisely such a limitation, providing a principled way for identifying a forget set from more abstract forms of unwanted evidence (Step 3)

be able to execute MU methods, one needs to preliminarily *identify the forget set based on user's unwanted evidence* (Step 3 of Fig. 1). This is a crucial and challenging task, which, to the best of our knowledge, has never been tackled so far.

1.2 State-of-the-art advancement

In this work, we fill this important gap, and study for the first time what we call the Forget-Set Identification problem (for short, ForSId): find the forget set D_f, such that its removal from the original training set D ensures the unlearning from an existing model M_D trained on D of all user-specified unwanted evidence D_u, while preserving the predictions for organization's desired cases D_w. This way, ForSId identifies the samples to be removed by analyzing model's predictions for both unwanted and wanted examples.

As such, ForSId is an important complementary problem to MU, to be used coupled with (and not in substitution for) MU methods. Specifically, ForSId serves as a foundational step to be performed prior to executing MU methods, allowing for extending the scope of MU to scenarios where directly providing the training samples to be removed is not possible or too difficult.

1.3 Applications

The proposed ForSId problem finds natural application in several real-world contexts. For instance, consider an end-user's service based on an image classification system. Suppose that the service provider is receiving a request to forget specific images (e.g., images of individuals who want to exercise their right to be forgotten, or outdated images). The user who performs such a forgetting request likely does not have access to the training set, for privacy reasons or simply because that training set is proprietary to the provider and/or it has been

anonymized. Thus, the user cannot provide a specific forget set to perform the unlearning task directly. Instead, the user can provide unlearning evidence in a more abstract form, e.g., examples of self-images (not included in the training data) that indicate broadly what needs to be unlearned, so that identifying a proper forget set corresponding to the desired unlearning evidence will be deferred to our ForSId problem the task.

A similar scenario arises in the context of AI systems trained on copyrighted artworks. Here, an artist may request that the model forget their style or specific artworks. However, they often do not have access to the exact training data used, nor do they know precisely which of their works were included. Instead, they can only provide examples of their art as unlearning evidence, requiring the system to identify which training samples correspond to this evidence to ensure compliance with the forgetting request.

More generally, the limitation of requiring a forget set arises in many MU scenarios where the exact data to be removed is unknown, whereas the undesired behavior is well understood. For instance, when regulations or other forms of sensitive documents need to be discarded from natural-language processing systems despite the lack of precise knowledge about which training data contributed to their learning.

1.4 Technical contributions

We provide a combinatorial-optimization formulation of the ForSId problem and show its NP-hard ness through a reduction from the Knapsack problem (Kellerer et al., 2004). We then establish a connection between ForSId and Red-Blue Set Cover (RBSC) (Caprara et al., (2000), a variant of the well-known Set Cover problem. Particularly, we devise a novel, weighted-coverage-based formulation of RBSC, which is dubbed Red-Blue ForSId and shown to constitute an effective and efficient proxy for the original ForSId problem. In formulating Red-Blue ForSId, a major technical challenge that we achieve is to come up with a notion of coverage that can effectively express the desiderata of ForSId. Defining the Red-Blue ForSId problem mainly serves the purpose of facilitating algorithm design for the ForSId problem. In fact, the ultimate algorithm we propose to tackle ForSId, termed ForSId-via-RBSC, consists of solving the proxy Red-Blue ForSId problem.

In this paper, we introduce a novel research line in the MU landscape that has not been considered so far. To support further development of this area, we release a codebase,[1]

1.5 Summary and roadmap

To summarize, in this work:

- We study for the first time Forget-Set Identification (ForSId), i.e., the problem of identifying a forget set D_f as a subset of a training set D utilized to train a model M_D whose removal from D allows for unlearning unwanted evidence D_u from M_D and keeping wanted evidence D_w.
- We formalize ForSId and show its NP-hard ness (Sect. 3).
- We establish a connection between ForSId and Red-Blue Set Cover, and exploit it to

[1] https://anonymous.4open.science/r/FSI—Forget-Set-Identification-F78C. which contains all the code for full reproducibility of methods and experiments of this paper, along with an easily extensible framework which allows researchers and practitioners to seamlessly integrate their own approaches.

design the ForSId-via-RBSC algorithm for ForSId (Sect. 4).
- We provide extensive experiments on a variety of datasets and settings, and involving a number of baselines (Sect. 5). Results attest the high performance of the proposed ForSId-via-RBSC algorithm, and its superiority over the baselines (Sect. 6).

Sect. 2 overviews the related work. Section 7 concludes the paper.

2 Related work

2.1 Machine unlearning (MU)

Since its introduction (Cao & Yang, 2015), MU has been a very active area of research. A plethora of different MU approaches have been devised, either exact or approximate, and adhering to various principles, including data reorganization and model manipulation (Xu et al., 2024), and reproducibility-oriented frameworks (D'Angelo et al., 2025). There also exist (recent) works that investigate how the properties of the forget set render the MU process more or less difficult (Fan et al., 2024; Zhao et al., 2024).

Our work is orthogonal to the bulk of the MU literature. MU approaches take a forget set as input (either explicitly or implicitly, e.g., by deriving it as the difference of the training set and an input retain/preservation set (Bonato et al., 2024; Newatia et al., 2024)). Instead, we tackle the (so far unstudied) preliminary problem of identifying a forget set from the user's unlearning evidence. As such, the problem and algorithms defined in this work complement MU methodologies, although experimentally testing them on the reliable exact unlearning task is the most grounded way to assess their impact (cf. Section 5).

Variants of MU include the *selective forgetting* problem (Shibata et al., 2021), whose goal is to forget one or more entire classes from a trained model, rather than individual samples. This is a goal that is also at the basis of other slightly different, but still class-level forgetting problems (Kuwana et al., 2024; Tarun et al., 2024; Ye et al., 2022). *Model editing* (or *knowledge editing*) has recently emerged in the large language model (LLM) landscape as the problem of modifying an LLM to incorporate specific knowledge, without negatively influencing other irrelevant knowledge (Wang et al., 2025).

Although similar in spirit, those variants remain anyway different from MU. The proposed ForSId problem is mainly aimed at being auxiliary in a MU context.

2.2 Training data influence

Several existing works tackle the problem of assessing the "influence" (or "impact") of individual training samples on the prediction of some given test samples. Approaches of this kind are based on the Fisher information matrix (Foster et al., 2024), influence functions (Koh & Liang, 2017), or similarity metrics, such as the Mahalanobis inner product (Kung, 2014; Cadamuro et al., 2016). As for deep learning models, advanced techniques such as representer points (Yeh et al., 2018) offer a way to measure the influence of multiple samples simultaneously.

In this work, we use training data influence methods as a building block of the algorithm we design for our ForSId problem (cf. Sect. 4).

Red-Blue Set Cover (RBSC) is variant of the well-known Set Cover problem, where two finite sets of "red" (R) and "blue" (B) elements are given, along with a family $\mathcal{S} \subseteq 2^{R \cup B}$ of subsets, and the objective is to select a subfamily $\mathcal{C} \subseteq \mathcal{S}$ that covers all blue elements while minimizing the number of covered red elements (Carr et al., 2000). The literature in RBSC has mostly focused on analyzing its theoretical properties, studying the (in)approximability of both the basic formulation (Carr et al., 2000; Chlamtác et al., 2023; Elkin & Peleg, 2007) and geometric variants of it (Abidha & Ashok, 2024; Ashok et al., 2017; Chan & Hu, 2015; Madireddy & Mudgal, 2023).

In this work, we use RBSC as a tool to take inspiration from for algorithm design for the proposed ForSId problem. Advancing the state of the art in RBSC is beyond the scope of this work.

3 Problem definition

We are given a machine-learning model M_D trained on a training set D. The machine-learning task underlying M_D is left unspecified in our problem: any task can in principle be handled, as long as the notions and functions that are left general in the definition of the problem are properly instantiated to account for the desired task (see next). We assume D to correspond to a set of pairs $\{(\mathrm{t}_i, y_i)\}_{|D|}$, where each $\mathrm{t}_i \in \mathbb{R}^d$ is a d-dimensional real-valued numerical vector representing the i-th training set sample, and y_i is the corresponding ground-truth label. The various y_i's – as well as the output $M_D(\mathrm{t})$ of the M_D model applied to any input t – take values from a domain $\mathcal{Y}$, which is instantiated based on the specific machine-learning task. For instance, in case of (univariate) regression $\mathcal{Y} = \mathbb{R}$, for single-label binary classification $\mathcal{Y} = \{0, 1\}$, for multilabel multiclass classification $\mathcal{Y} = \mathbb{N}^k$, $k \geq 1$. Let also:

- $D_f \subseteq D$ be a *forget set*, i.e., a subset of the training set that needs to be discarded in order to meet the unlearning desiderata specified by the user through an "unwanted set" (see next).
- $M_{D \setminus D_f}$ be the machine-learning model obtained by retraining the original input model on the training set $D \setminus D_f$, often referred to as *gold model* in the Machine Unlearning literature (Chundawat et al., 2023).
- D_u be an *unwanted set*, i.e., a set $\{\mathrm{u} \in \mathbb{R}^d\}$ of evidence (that satisfies the model input) that constitutes what the user requires to be unlearned from M_D. In general, $D_u \cap D$ may be nonempty. If this is the case, the samples in $D_u \cap D$ are directly included in D_f. We say that a retrained model $M_{D \setminus D_f}$ has actually unlearned a evidence u if the output of $M_{D \setminus D_f}$ on u differs from the output of M_D on u by at least a certain user-defined threshold $T > 0$, i.e., $|M_{D \setminus D_f}(\mathrm{u}) - M_D(\mathrm{u})| \geq T$. The threshold T is set according to the machine-learning task and the application domain. E.g., in single-label classification (either binary or multiclass), $T = 1$, as any u is reasonably considered unlearned if its new predicted class is other than the originally predicted one. In regression tasks, T is set so that changes in the numerical predictions that exceed T are large enough to recognize the predictions as actually changed. Note that the threshold T plays the role of quantifying to what extent the various unwanted samples have been actually forgotten. The forgetting on the unwanted samples is counterbalanced by the desideratum of keep-

ing the original model predictions unchanged on a set of wanted instances according to a properly-defined prediction similarity function (see next)

- D_w be a *wanted set*, i.e., a set $\{\mathrm{w} \in \mathbb{R}^d\}$ of evidence (that satisfies the model input) on which the prediction of the retrained model (resulting from the unlearning process) is required to be as much as possible equal to the prediction of the original model. We assume $D_w \cap D_u = \emptyset$, as an evidence can clearly exhibit wanted or unwanted behavior only, not both.
- σ be a *prediction similarity function*, i.e., a real-valued function which takes the original model M_D, the model $M_{D \setminus D_f}$ retrained on $D \setminus D_f$, and the wanted set D_w as arguments, and returns a real number expressing the similarity between the outputs of M_D and $M_{D \setminus D_f}$ on the various samples $\mathrm{w} \in D_w$. Like T, σ is defined based on the machine-learning task and the application. As an example, in single-label binary classification, σ might be defined as the number of samples of D_w for which M_D and $M_{D \setminus D_f}$ give the same classification, i.e., $\sigma := \sum_{\mathrm{w} \in D_w} 1 - |M_D(\mathrm{w}) - M_{D \setminus D_f}(\mathrm{w})|$. In single-label multiclass classification, σ might correspond to the average cross-entropy across all the samples of D_w using the output of M_D as a ground-truth.

Informally speaking, in this work we deal with the problem of identifying a forget set whose removal from the original training set leads to both unlearning all the evidence in a given unwanted set and maximum similarity in the model's outputs on the wanted set before and after the unlearning process. We formalize such desiderata in the following novel combinatorial-optimization problem:

Problem 1 *Given a training set D, a machine-learning model M_D trained on D, an unwanted set D_u, a wanted set D_w, a prediction similarity function σ, and a threshold $T > 0$, find a forget set $D_f^* \subseteq D$ that maximizes σ subject to the constraint that all the samples of D_u get unlearned:*

$$D_f^* = \underset{D_f \subseteq D}{argmax} \;\; \sigma(M_D, M_{D \setminus D_f}, D_w) \tag{1}$$

$$subject to |M_{D \setminus D_f}(\mathrm{u}) - M_D(\mathrm{u})| \geq T, \forall \mathrm{u} \in D_u. \tag{2}$$

The rationale of Problem 1 is as follows. The target scenario of properly identifying a forget set comes with two main desiderata. On the one hand, as Desideratum 1, the unwanted samples D_u need to be "forgotten" as much as possible (where "forgotten" means that the prediction on each sample in D_u after the unlearning process should considerably change with respect to the original prediction). On the other hand, as Desideratum 2, we also need to ensure that the original predictions on the wanted samples D_w are, to the largest possible extent, similar to the original predictions achieved before the unlearning. These two desiderata are clearly conflicting to each other. In order to formalize such desiderata as objectives of an optimization problem, we adopt the usual way of letting one desideratum to correspond to the objective function of the resulting optimization problem, and putting the other objective as a constraint. In our context, and in machine unlearning in general, the emphasis is on forgetting the unwanted samples. Therefore, in the formal definition of our ForSId problem, we decided to model Desideratum 1 as a constraint and Desideratum 2 as

the objective function. Based on this, it is apparent that a scenario where forcing to change the prediction of a sample in D_u leads to a change in the prediction of other samples not in D_u is hindered by the objective function of the problem.

The ForSId problem fills an important gap in the MU landscape, addressing the so-far overlooked challenge of identifying a forget set out of user's unlearning desiderata. Once a forget set is obtained, it can be used to perform the actual unlearning by resorting to existing MU methods. As such, ForSId constitutes an essential intermediate step that allows for extending the range of applicability of MU to all those settings where user's unlearning evidence does not correspond to (or is too hard to be directly expressed in terms of) a forget set. This arises in many real-world scenarios, such as users' rights compliance, personalized recommender systems, and regulation update (cf. Introduction). The main notations used in the paper are summarized in Table 1.

Theorem 1 *Problem 1 is* NP*-hard.*

Proof We reduce from the well-established NP-hard Knapsack problem (Kellerer et al., 2004): given a constant $C > 0$ and a set X of items, where each $x_i \in X$ is assigned value $v_i > 0$ and weight $w_i > 0$, find a subset $X^* \subseteq X$ of items that maximizes the total value $\sum_{x_j \in X^*} v_j$ subject to the constraint $\sum_{x_j \in X} w_j \leq C$. Given an instance $I = \langle C, X, \{v_i\}, \{w_i\}\rangle$ of Knapsack, we construct an instance $I' = \langle D, M_D, D_u, D_w, \sigma, T\rangle$ of ForSId as follows:

- $D = X$;
- M_D is set to a constant function that always outputs H, regardless of the input instance x_i, where H is an arbitrary positive constant, i.e., $M_D(x_i) = H, \forall x_i \in D$;
- D_u is composed of a single sample, while $D_w = \emptyset$;
- $\sigma(M_D, M_{D\setminus D_f}, D_w) = \sum_{x_i \in D_f} v_i$;
- $T = W$, where $W = \sum_{x_i \in X} w_i$ is the sum of the weights of all the items.

Moreover, for any $D_f \subseteq D$, we set $M_{D\setminus D_f}$ to a constant function that always outputs $H + W + (C - \sum_{x_i \in D_f} w_i)$, regardless of the input instance x_i, i.e., $M_{D\setminus D_f}(x_i) = H + W + (C - \sum_{x_i \in D_f} w_i)$, $\forall x_i \in D$. The construction of such an instance I' can clearly be accomplished in polynomial time.

Table 1 Main notations used in this paper

Symbol	Description
$D = \{(\mathrm{t}_i, y)\}_{\lvert D\rvert}$	Training set
M_D	Machine-learning model trained on D
$D_f \subseteq D$	Forget set
$M_{D\setminus D_f}$	Machine-learning model retrained on $D \setminus D_f$
$D_u = \{\mathrm{u}\}_{\lvert D_u\rvert}$	Unwanted set
T	Positive threshold to recognize any $\mathrm{u} \in D_u$ as unlearned
$D_w = \{\mathrm{w}\}_{\lvert D_w\rvert}$	Wanted set
$\sigma(M_D, M_{D\setminus D_f}, D_w)$	Prediction similarity function
$\alpha(M_D, \mathrm{t}, \mathrm{z}) \in \mathbb{R}$	Impact of $\mathrm{t} \in D$ on the prediction $M_D(\mathrm{z})$ of $\mathrm{z} \in D_u \cup D_w$ by model M_D

The objective function of ForSId on instance I' is $\text{argmax}_{D_f \subseteq D}\, \sigma(M_D, M_{D\setminus D_f}, D_w) = \text{argmax}_{X' \subseteq X} \sum_{x_i \in X'} v_i$, which corresponds to Knapsack 's objective function on instance I. As far as ForSId 's constraint (Eq. (2)), instead, it holds that:

$$
\begin{aligned}
&|M_{D\setminus D_f}(\mathrm{u}) - M_D(\mathrm{u})| \geq T, \forall \mathrm{u} \in D_u \\
&\Leftrightarrow |M_{D\setminus D_f}(\mathrm{u}) - M_D(\mathrm{u})| \geq T, \text{on the only } \mathrm{u} \in D_u \\
&\Leftrightarrow |H + W + (C - \textstyle\sum_{x_i \in D_f} w_i) - H| \geq W \\
&\Leftrightarrow W + C - \textstyle\sum_{x_i \in D_f} w_i \geq W \;\Leftrightarrow\; \textstyle\sum_{x_i \in D_f} w_i \leq C,
\end{aligned}
$$

where the absolute-value removal in the second last implication holds because $W + C - \sum_{x_i \in D_f} w_i \geq 0$.

All in all, solving ForSId on instance I' corresponds to finding a subset $D_f^* = X^* \subseteq X = D$ that maximizes $\sum_{x_i \in D_f^* = X^*} v_i$ subject to the constraint $\sum_{x_i \in D_f^* = X^*} w_i \leq C$, which corresponds to solving Knapsack on instance I. The theorem follows. □

The above theorem states that the proposed ForSId problem is at least as difficult as Knapsack. However, the ForSId instance constructed in the reduction is a simplistic one (it does not use D_u or D_w, and employs constant functions as models M_D and $M_{D\setminus D_f}$). Intuitively, this makes us think that ForSId is more challenging than Knapsack to be tackled in practice. This in turn means that the connection to Knapsack is not really helpful in terms of algorithm design. In fact, the algorithm we devise for ForSId is based on different technical insights (cf. next section).

4 Algorithms

Connection to RBSC **.** Our ForSId problem can be interpreted in terms of Red-Blue Set Cover (RBSC) (cf. Sect. 2). The set B of blue elements (the ones to be necessarily "covered") corresponds to the unwanted set D_u. The set R of red elements (the ones to be "covered" the least possible) corresponds to the wanted set D_w. The input family $\mathcal{S}$ of subsets – which the ultimate desired forget set D_f is selected from – corresponds to the set 2^D of all possible subsets of the training set D. That is the easy part. What is challenging is to come up with a proper definition of what "coverage" means here. We discuss this next.

"Coverage" of D_u and D_w. In principle, recognizing a certain set $D_f \in \mathcal{S} = 2^D$ as covering elements of D_u or D_w would require retraining the original model (over a training set $D \setminus D_f$). In fact, according to ForSId 's problem statement (Problem 1), $M_{D\setminus D_f}$ needs to be evaluated on both D_w (Eq. (1)) and D_u (Eq. (2)). Even if algorithms are designed to limit the number of subsets of D that need to be generated and explored during execution, retraining a machine learning model for each of these subsets remains complex and inefficient. Resorting to MU methods to unlearn any of such subsets of D (sacrificing optimality) would make things better, but not that much. For this reason, here we define our desired notion of coverage in terms of the impact that training samples exhibit over the predictions of a trained machine-learning model, without requiring any model retraining. Specifically, for each training sample $\mathrm{t} \in D$, and each sample $\mathrm{z} \in D_u \cup D_w$ of the (un)wanted set, let $\alpha(M_D, \mathrm{t}, \mathrm{z}) \in \mathbb{R}$ denote the impact of t on the prediction $M_D(\mathrm{z})$ of z by model M_D: the higher $\alpha(M_D, \mathrm{t}, \mathrm{z})$, the more responsible t is for the prediction assigned to z by M_D, and

vice versa. In other words, the higher $\alpha(M_D, \mathrm{t}, \mathrm{z})$, the more likely is that retraining the model on $D \setminus \{\mathrm{t}\}$ will lead to a change in the prediction of z, and vice versa.

The α scores can be computed in various ways, such as through similarity between samples, or data-influence measures (Ly et al., 2017; Foster et al., 2024; Koh & Liang, 2017; Kung, 2014; Cadamuro et al., 2016; Yeh et al., 2018) (cf. Sect. 2).

The Red-Blue ForSId problem. We translate the above principles into an optimization problem which is a variant of RBSC and is dubbed Red-Blue ForSId. For reasons that will become clear in a while, we provide an integer linear programming (ILP) formulation:

Problem 2 *Given a training set D, let each sample $\mathrm{t} \in D$ be assigned a decision binary variable $b_\mathrm{t} \in \{0, 1\}$ expressing whether t is part ($b_\mathrm{t} = 1$) or not ($b_\mathrm{t} = 0$) of the output forget set. Given a training set D, a machine-learning model M_D trained on D, an unwanted set D_u, a wanted set D_w, a threshold $T > 0$, and impact scores $\alpha(M_D, \mathrm{t}, \mathrm{z})$, $\forall \mathrm{t} \in D$, $\forall \mathrm{z} \in D_u \cup D_w$,*

$$minimize \sum_{\mathrm{t} \in D, \mathrm{w} \in D_w} b_\mathrm{t} \ \alpha(M_D, \mathrm{t}, \mathrm{w}) \tag{3}$$

$$subject to \sum_{\mathrm{t} \in D} b_\mathrm{t} \ \alpha(M_D, \mathrm{t}, \mathrm{u}) \geq T, \ \forall \mathrm{u} \in D_u, \tag{4}$$

$$b_\mathrm{t} \in \{0, 1\}, \ \forall \mathrm{t} \in D. \tag{5}$$

Problem 2 selects a subset $D_f = \{\mathrm{t} \in D \mid b_\mathrm{t} = 1\}$ (forget set) of the input training set D so that the cumulative impact of the selected training samples on all the wanted samples is minimized (Eq. (3)), subject to the constraint that the cumulative impact of the selected training samples on each unwanted sample is large enough, i.e., no less than threshold T (Eq. (4)). Conceptually speaking, threshold T in Problem 2 plays a similar role to the threshold used in Problem 1. However, instead of affecting actual changes in predictions like in Problem 1, threshold T in Problem 2 affects impact estimates α. Here, we slightly abuse of notation, as we denote with the same symbol T for the threshold in both Problem 1 and Problem 2. Unless otherwise specified, in the remainder of the paper, when referring to T we mean the threshold in Problem 2. Problem 2 can be easily recognized as a variant of the basic RBSC problem, where the blue and red sets correspond to the unwanted and wanted sets, respectively. The main differences with respect to the basic RBSC are: (*i*) the input family is implicitly defined as all the possible subsets of the training set, and (*ii*) a notion of weighted coverage is used (expressed as the cumulative impact of the various selected training samples).

Input: Training set D; machine-learning model M_D trained on D; unwanted set D_u; wanted set D_w; threshold $T > 0$
Output: Forget set $D_f \subseteq D$

1: Compute impact scores $\alpha(M_D, \mathbf{t}, \mathbf{z})$, $\forall \mathbf{t} \in D$, $\forall \mathbf{z} \in D_u \cup D_w$
2: Build linear program P of Problem 2 based on $\alpha(M_D, \mathbf{t}, \mathbf{z})$ and T
3: $\{b_\mathbf{t} \mid \mathbf{t} \in D\} \leftarrow$ ILPSolver(P)
4: $D_f \leftarrow \{\mathbf{t} \in D \mid b_\mathbf{t} = 1\}$

Algorithm 1 ForSId-via-RBSC

4.1 The proposed ForSId-via-RBSC algorithm

The forget set ultimately selected by solving the Red-Blue ForSId problem (Problem 2) will contain training samples whose removal from D leads to the least possible change in the prediction of the wanted samples (as quantified by the impact scores α) and to a highly likely change of prediction of each unwanted sample (as quantified by the impact scores α exceeding T). This way, Red-Blue ForSId 's objective function (Eq. (3)) and Red-Blue ForSId 's main constraint (Eq. (4)) represent reasonable proxies for ForSId 's objective function (Eq. (1)) and ForSId 's main constraint (Eq. (2)), respectively. Note that the use of such proxies eliminates the dependence on a prediction similarity function σ. This side effect is desirable, as it avoids defining an ad-hoc σ for any machine-learning task and application context, which may be otherwise challenging.

Motivated by this, the algorithm we propose for our ForSId problem consists in solving the proxy Red-Blue ForSId problem. At first glance, as Red-Blue ForSId is a variant of RBSC, an option to solve it would be to resorting to existing algorithms for RBSC (cf. Sect. 2). However, all these algorithms are mostly theoretical (and, often, not really easy-to-implement either). They are mainly aimed at proving (in)approximability properties, rather than being profitably used in practice (in fact, no experimental evaluation has ever been carried out on such algorithms). Moreover, our notion of weighted coverage makes the adaptation of those algorithms to our Red-Blue ForSId variant challenging (and, perhaps, any adaptation would make them lose their approximation guarantees). For all these reasons, here we propose to solve the Red-Blue ForSId problem with an ILP solver. We defer to future work the design of different and more specialized algorithms. The resulting proposed algorithm is termed ForSId-via-RBSC and outlined in Algorithm 1.

4.2 Complexity

The running time of ForSId-via-RBSC is dominated by the time taken by the ILP solver. The other steps cost as follows. Computing the impact scores (Line 1) depends on the specific technique employed. However, it takes at least time $\Omega(|D| \times (|D_u| + |D_w|))$, as it yields a score $\forall \mathrm{t} \in D, \forall \mathrm{z} \in D_u \cup D_w$. Building the linear program (Line 2) takes $\mathcal{O}(|D| \times (|D_u| + |D_w|))$ time, while constructing the output D_f (Line 4) takes $\mathcal{O}(|D|)$ time.

The space complexity of ForSId-via-RBSC is $\mathcal{O}(|D| \times (|D_u| + |D_w|))$, which corresponds to both the number of α scores and the size of the linear program.

5 Experimental setting

5.1 Datasets (Table 2)

We selected datasets that are used in the bulk of the literature in MU (Chundawat et al., 2023; Golatkar et al., 2020; Foster et al., 2024). These include the well-established CIFAR family of datasets, namely CIFAR-10, CIFAR-20, and CIFAR-100 (Krizhevsky & Hinton, 2010). The CIFAR family is composed of 60k images of 32x32 color pixels. CIFAR-10 consists of 10 distinct classes, whereas CIFAR-100 includes both coarse-grained labels (forming CIFAR-20) and fine-grained labels (defining CIFAR-100). We chose CIFAR-10 because

Table 2 Characteristics of the datasets used in the experiments, along with the accuracy achieved by a machine-learning model whose network model is reported under brackets (CL: convolutional layers, FC: fully-connected layers)

Dataset	Features	Type	#Samples	#Classes	Accuracy [network model]
CIFAR-10	32x32x3	Image	60 000	10	$\sim 95\%$ [3 CL (64,128,256) + 3 FC]
CIFAR-20	32x32x3	Image	60 000	20	$\sim 85\%$ [3 CL (64,128,256) + 3 FC]
CIFAR-100	32x32x3	Image	60 000	100	$\sim 65\%$ [VGG-16]
Wine Quality	11	Tabular	4 898	10	$\sim 65\%$ [64,32,32 FC]

it has clear decision boundaries within its classes. Instead, CIFAR-20 and CIFAR-100 present high uncertainty and overlapping intra-class decision boundaries (testified by the decreasing accuracy values reported in Table 2). As such, CIFAR-20 and CIFAR-100 constitute a highly-challenging testbed. For each experiment with those datasets, we selected a combination of 10 distinct coarse-grained labels for CIFAR-10 and CIFAR-20, whereas, for CIFAR-100, we similarly selected from the fine-grained labels.

Additionally, to the CIFAR family, we incorporated the Wine Quality dataset (Cortez et al., 2009) to assess our method on a smaller-scale (11 features and around 5k samples), multi-class tabular dataset, where labels represent quality scores ranging from 0 to 10. We use the Wine Quality dataset in its standard classification form, as provided by the UCI Repository (Cortez et al., 2009). As for CIFAR-10 and 20 we sampled 1 000 samples for the unwanted set D_u and 1000 samples for the wanted set D_w. In the case of CIFAR-100 and Wine, as 1 000 instances per class are not available, we selected 100 for both the unwanted and wanted sets.

5.2 Baselines

The ForSId problem we tackle in this work is a newly formulated one. As such, no established methods or benchmarks exist for it in the literature. For this reason, here we design well-founded yet nontrivial baseline approaches to be involved for comparison to the proposed ForSId-via-RBSC algorithm. One of such baselines operates in the data space, while the remaining ones exploit the same training data influence function as our ForSId-via-RBSC. We describe these baselines next.

K-nearest neighbors (k-NN): For each sample $\mathrm{u} \in D_u$ of the unwanted set D_u, this baseline computes u's *k*-nearest neighbors in the training set D using some distance measure $dist(\mathrm{u}, \mathrm{t})$ in the data space. The forget set D_f is ultimately computed as the union of all such *k*-nearest neighbors. We chose $dist(\mathrm{u}_i, \mathrm{t}_j) = \|\mathrm{u}_i - \mathrm{t}_j\|_2$ as the distance function due to its computational efficiency and widespread use in k-NN searches and image datasets. This baseline aims to identify the closest instances within the data space to assess whether the problem can be effectively addressed through a very simple approach that completely discard the input model M_D.

Top-K Union (TopK-U) computes the forget set D_f as the union of the *k* most influential samples t of the training set D for each sample u in the unwanted set D_u, according to the chosen impact score function α, i.e., $D_f = \bigcup_{\mathrm{u} \in D_u} \mathrm{TopK}(M_D, D, \mathrm{u}, k)$, where $\mathrm{TopK}(M_D, D, \mathrm{u}, k) = \sum_{\mathrm{t} \in S_k} \alpha(M_D, \mathrm{t}, \mathrm{u})$, and $S_k \subseteq D$ is the set of the *k* training samples exhibiting the highest α score.

Note that TopK-U solely leverages the impact of the training samples on the unwanted set. To ensure a fair comparison, this baseline (and the next ones) adopt the same impact score function α, which constitutes their key strength.

Top-K Difference (TopK-D) computes the forget set D_f as the set difference between the union of the k most influential training samples $\mathrm{t} \in D$ for each sample $\mathrm{u} \in D_u$ and the union of the k most influential training samples $\mathrm{t} \in D$ for each sample $\mathrm{w} \in D_w$, based on the chosen impact score function α, i.e., $D_f = \bigcup_{\mathrm{u} \in D_u} \mathrm{TopK}(M_D, D, \mathrm{u}, k) \Big\setminus \bigcup_{\mathrm{w} \in D_w} \mathrm{TopK}(M_D, D, \mathrm{w}, k)$.

Impact Disparity (ImpDisp) computes the forget set D_f as the set of training samples $\mathrm{t} \in D$ for which the disparity between their cumulative impact on $\mathrm{u} \in D_u$ and $\mathrm{w} \in D_w$ exceeds a given threshold l, i.e., $D_f = \left\{ \mathrm{t} \in D \middle| \left(\sum_{\mathrm{u} \in D_u} \alpha(M_D, \mathrm{t}, \mathrm{u}) - \sum_{\mathrm{w} \in D_w} \alpha(M_D, \mathrm{t}, \mathrm{w}) \right) > l \right\}$.

If and only if the threshold l is set to 0, as in our case, the impact disparity admits the following equivalent optimization formulation:

$$\mathrm{maximize}_{\mathrm{t} \in D} \sum_{\mathrm{u} \in D_u} \alpha(M_D, \mathrm{t}, \mathrm{u}) - \sum_{\mathrm{w} \in D_w} \alpha(M_D, \mathrm{t}, \mathrm{w}) \tag{6}$$

5.3 Impact score function α

As stated in Sect. 4, α scores can be computed in various ways, such as via similarity among samples, data-influence measures (Ly et al., 2017; Foster et al., 2024; Koh & Liang, 2017; Kung, 2014; Cadamuro et al., 2016; Yeh et al., 2018), or the Fisher information matrix (Foster et al., 2024; Lev & Wilson, 2024; Liu et al., 2023). In this work, we adopt the Representer Point framework (Yeh et al., 2018) to instantiate the impact scores $\alpha(M_D, \mathrm{t}, \mathrm{z})$. This is mainly motivated by the fact that representer points are particularly well-suited for our problem since the total impact of all training samples on a prediction is decomposed into a sum of individual contributions – a key property that aligns well with the Red-Blue ForSId formulation.

Assessment criteria. To evaluate the effectiveness of ForSId-via-RBSC and the baseline methods, it is essential to assess their ability to successfully unlearn the unwanted set D_u while preserving the predictions on the wanted one D_w. To achieve this, we define a set of benchmarking measures based on $M_{D \setminus D_f}$ – the machine learning model obtained by retraining the original input model on $D \setminus D_f$. Note that we evaluate our method and the baselines w.r.t. the exact unlearning task (i.e., from-scratch retraining) to avoid any confounding factors of inexact unlearning methods.

Specifically, we employ the following measures.

Efficacy, defined on top of a revised confusion matrix where the true positives TP (wanted unchanged) and false negatives FN (wanted changed) correspond to the number of predictions of the samples in the wanted set D_w that have unchanged/changed in the unlearned model $M_{D \setminus D_f}$ with respect to the original model M, respectively. Conversely, the false positives FP (unwanted unchanged) and true negatives TN (unwanted changed) correspond to the number of predictions of the samples in the unwanted set D_u that have unchanged/changed in the unlearned model $M_{D \setminus D_f}$ with respect to the original model M, respectively.

Wanted Efficacy (WE), defined as the ratio of wanted samples that maintain their original predictions relative to the total number of wanted sets, i.e., $\frac{TP}{TP+FN}$.

Unwanted Efficacy (UE), defined in a similar way to WE, it is the ratio of unwanted samples whose predictions changed to the total number of unwanted samples, i.e., $\frac{TN}{FP+TN}$.

Balanced Efficacy (BE), defined as the mean between UE and WE, i.e., $2 \cdot \frac{FP \cdot FN}{TP \cdot TN}$.

Accuracy Loss (Acc. Δ). To assess the impact of the forget set computed by each method in a broader context, we also consider the accuracy divergence between M and $M_{D \setminus D_f}$ on a test set D_t disjoint from all the other sets at hand (i.e., D, D_u, D_w). The rationale is that an effective forget set should induce a maximal change in the unwanted samples' predictions while minimizing the retrained model's overall accuracy loss.

Overall Effectiveness (H). To provide the reader with a comprehensive picture of the wanted and unwanted efficacy with the accuracy loss in an all-in-one score, we compute the harmonic mean of the WE, UE, and Acc. Δ measures, i.e., $\frac{3}{\frac{1}{UE}+\frac{1}{WE}+\frac{1}{\Delta Acc}}$.

Forget set size (FS-Size). We report the size of the identified forget set for each method as the average number of samples removed, providing a quantitative measure of the extent of forgetting.

Running Time (RunT) of any method involved in the comparison, measured in seconds.

5.4 Parameters and other settings

As for the baselines using parameter *k* (namely k-NN, TopK-U, and TopK-D), we selected *k* ranging from 5 to 30 000. Specifically, we used a progressively increasing sequence, starting with smaller increments (e.g., $k = 5, 10, 20, 30, 50$) and transitioning to larger gaps at higher values (e.g., $k = 100, 150, 200, 1\,000$), ensuring broader coverage across different scales. As we cannot guarantee monotonic behavior for our method or the baselines with respect to their parameter, we adopt a linear tuning approach to ensure a more robust and comprehensive analysis. We also systematically investigate the behavior of the methods as the parameter *k* approaches the size of the full dataset, in order to stress-test the methods and reveal asymptotic trends (this study is shown in-depth in Sect. 8.2). As for our ForSId-via-RBSC, we tuned parameter *T* ranging from $1e{-}5$ to 50, using a finer granularity in the empirically best-performing range (10 to 20). As a neural networks model, we used VGG-like architecture for Image-based classification, while for tabular classification, we used a feed-forward fully connected neural network with three hidden layers (64, 64, 32). In both cases, we trained the networks with the Adam optimizer for 30 epochs with a learning rate of 0.01. All networks were trained to achieve $> 95\%$ training accuracy. The last layer was fine-tuned to ensure that the predictions closely aligned with the representer theorem formulation ($> 99\%$ Pearson correlation between ground truth prediction and prediction by representer theorem, as described in Yeh et al. (2018)). All experiments were run 10 times, with each run sampling the Unwanted Set D_u from a different class (classes 0 to 10, and random classes for CIFAR-100). The test set D_t was kept the same across all runs, comprising 20% of the original dataset.

5.5 Testing environment

For timed experiments, we run all methods on a Rocky Linux 8 server with 32-core Intel(R) Xeon(R) CPU @ 2.30GHz, up to 64GB RAM, NVIDIA A100 (MIG 56 sm 40GB) GPU.

6 Experimental results

In the following, we provide and discuss the experimental results of the proposed ForSId-via-RBSC algorithm (Algorithm 1), and compare them to the baselines described in Sect. 5. In Appendix 8.1, we report the tables with the standard deviations, here omitted for the sake of space. In Appendix 8.2, we report parameter studies on parameters T and K, and study the behavior of our method and baselines when they grow to the limit of the dataset size.

We conducted extensive experiments designed to capture the real-world complexities of Forget-Set Identification, where the distinction between the wanted and unwanted evidence may not always be clear-cut. Specifically, we structure our evaluation around three different experimental settings, each characterized by varying degrees of uncertainty in the distinction between the unwanted and wanted sets. Moreover, the chosen real-world datasets (cf. Sect. 5) allow us to highlight the capabilities of our method in terms of generalization across different data distributions and characteristics, while also exposing potential limitations when faced with complex, overlapping decision boundaries between the wanted and unwanted sets. This experimental design aims to provide a rigorous and transparent evaluation of the ForSId-via-RBSC algorithm, demonstrating its empirical advantages over the baselines.

6.1 One vs others

This experiment is intended to have a good separation between the unwanted and wanted sets while keeping low variance in selecting unwanted evidence. We sample the unwanted set D_u from one class c and the wanted set D_w from all the remaining classes. This way, no sample from class c is part of the wanted set. The results of this experiment are reported in Table 3

6.2 One vs all

This experiment has a similar setting to the One vs others one, but here the wanted set D_w is sampled following the distribution of all classes, including c. This will cause a higher uncertainty since the samples of class c contribute to both wanted and unwanted sets. The results of this experiment are reported in Table 4.

6.3 Many vs many

In this last experiment, we pushed unwanted set's variance further. We sample the unwanted set D_u and the wanted set D_w from two disjoint sets of multiple classes. The results of this experiment are reported in Table 5.

Table 3 Comparison of ForSId-via-RBSC (Ours) with baselines methods

		BE ↑	UE ↑	WE ↑	Acc. Δ↓	H ↑	FS-Size	RunT (s) ↓
CIFAR-10	k-NN	0.605	0.554	0.657	0.191	0.639	36814.818	58.122
	TopK-U	0.639	0.345	0.932	0.018	0.585	10205.091	50.029
	TopK-D	0.525	0.055	**0.996**	**0.001**	0.148	613.182	**25.622**
	ImpDisp	0.655	0.522	0.788	0.122	0.676	24278.250	2658.045
	Ours	**0.982**	**0.999**	0.965	0.077	**0.952**	19266.000	1094.689
CIFAR-20	k-NN	0.767	0.608	0.927	0.011	0.795	7676.500	149.123
	TopK-U	0.843	0.798	0.889	0.014	0.878	8988.100	40.479
	TopK-D	0.661	0.322	**1.000**	**0.006**	0.584	1065.778	**31.510**
	ImpDisp	0.790	0.948	0.631	0.102	0.766	25296.900	3223.776
	Ours	**0.986**	**0.993**	0.979	0.011	**0.983**	5247.000	1226.710
CIFAR-100	k-NN	0.774	0.626	0.921	0.034	0.782	8308.800	20.272
	TopK-U	0.902	0.867	0.936	0.014	**0.913**	5098.500	**1.918**
	TopK-D	0.835	0.669	**1.000**	**0.007**	0.846	672.400	3.714
	ImpDisp	0.799	**0.968**	0.630	0.173	0.742	24618.100	322.440
	Ours	**0.968**	0.943	0.993	0.152	0.873	17023.667	187.575
Wine	k-NN	0.855	0.880	0.830	0.038	0.871	2508.200	**0.179**
	TopK-U	0.768	0.800	0.735	0.039	0.764	2186.250	1.326
	TopK-D	0.744	0.708	0.780	**-0.015**	0.735	115.500	0.239
	ImpDisp	0.775	0.761	0.790	0.021	0.763	2310.600	19.750
	Ours	**0.920**	**1.000**	**0.840**	0.028	**0.924**	1795.000	9.848

Benchmarking measures as the mean of 10 runs. Bold values are the best overall; underlined ones are second best. Efficacies (BE, UE, WE) are the most important measuresOne vs others

Table 4 Comparison of ForSId-via-RBSC (Ours) with baselines methods

		BE ↑	UE ↑	WE ↑	Acc. Δ↓	H ↑	FS-Size	RunT (s) ↓
CIFAR-10	k-NN	0.593	0.560	0.627	0.190	0.632	36660.100	54.281
	TopK-U	0.626	0.349	0.903	0.018	0.581	10137.400	50.680
	TopK-D	0.503	0.007	**0.999**	**− 0.001**	0.021	38.000	**6.135**
	ImpDisp	0.661	0.562	0.760	0.122	0.662	24621.300	2663.667
	Ours	**0.802**	**0.695**	0.910	0.033	**0.837**	10297.000	1281.942
CIFAR-20	k-NN	0.760	0.626	0.895	0.007	0.799	9828.250	48.670
	TopK-U	0.823	0.820	0.826	0.010	0.867	11907.875	36.859
	TopK-D	0.516	0.032	**0.999**	**− 0.004**	0.090	93.875	**5.903**
	ImpDisp	0.793	**0.948**	0.638	0.074	0.788	23874.500	2664.712
	Ours	**0.850**	0.864	0.835	0.022	**0.880**	6858.000	1237.051
CIFAR-100	k-NN	0.800	0.669	0.931	0.026	0.832	6963.625	22.702
	TopK-U	0.926	0.963	0.889	0.020	**0.936**	5156.375	**1.890**
	TopK-D	0.676	0.355	0.996	**0.001**	0.441	524.250	4.170
	ImpDisp	0.829	**1.000**	0.657	0.177	0.757	24636.125	330.794
	Ours	**0.988**	**1.000**	**0.977**	0.187	0.858	20417.667	112.443
Wine	k-NN	**0.797**	**0.820**	0.775	0.033	**0.832**	2368.500	**0.216**
	TopK-U	0.661	0.728	0.595	0.024	0.719	3875.000	3.324
	TopK-D	0.587	0.372	**0.802**	0.039	0.488	88.750	0.374
	ImpDisp	0.661	0.615	0.708	0.001	0.689	2457.750	25.294
	Ours	0.744	0.765	0.723	**− 0.029**	0.778	2319.000	11.209

Benchmarking measures as the mean of 10 runs. Bold values are the best overall; underlined ones are second best. Efficacies (BE, UE, WE) are the most important measuresOne vs all

Table 5 Comparison of ForSId-via-RBSC (Ours) with baselines methods

		BE ↑	UE ↑	WE ↑	Acc. Δ↓	H ↑	FS-Size	RunT (s) ↓
CIFAR-10	k-NN	0.552	0.476	0.628	0.216	0.586	39190.600	121.306
	TopK-U	0.579	0.327	0.831	0.072	0.559	27916.400	**75.401**
	TopK-D	0.543	0.093	**0.994**	**0.017**	0.233	6436.600	103.775
	ImpDisp	0.540	0.265	0.814	0.103	0.481	23792.000	3276.857
	Ours	**0.869**	**0.793**	**0.944**	0.262	**0.776**	30727.800	1065.329
CIFAR-20	k-NN	0.752	0.534	0.970	0.001	0.768	3838.429	46.738
	TopK-U	0.733	0.660	0.806	0.061	0.769	27496.143	**25.109**
	TopK-D	0.666	0.333	**1.000**	**0.009**	0.597	10531.600	25.686
	ImpDisp	0.539	0.370	0.709	0.065	0.567	23653.429	2717.737
	Ours	**0.945**	**0.895**	0.996	0.138	**0.849**	23344.200	1314.937
CIFAR-100	k-NN	0.722	0.485	0.958	**0.012**	0.727	6152.000	47.285
	TopK-U	0.800	0.787	0.813	0.189	0.755	28024.000	**13.336**
	TopK-D	0.709	0.421	**0.997**	0.104	0.653	15558.000	31.007
	ImpDisp	0.467	0.380	0.553	0.188	0.508	25987.000	2724.077
	Ours	**0.998**	**1.000**	**0.997**	0.256	**0.795**	24534.000	1756.515
Wine	k-NN	0.857	**1.000**	0.714	0.032	0.866	5196.000	**1.728**
	TopK-U	**0.921**	**1.000**	0.842	0.024	**0.927**	5194.000	59.013
	TopK-D	0.789	0.697	**0.882**	0.073	0.804	1853.000	23.323
	ImpDisp	0.752	0.658	0.845	0.014	0.804	2729.000	262.115
	Ours	0.801	0.752	0.851	-0.003	0.857	2638.000	124.070

Benchmarking measures as the mean of 10 runs. Bold values are the best overall; underlined ones are second best. Efficacies (BE, UE, WE) are the most important measuresMANY VS MANY

Result analysis. Across all experimental settings, ForSId-via-RBSC (referred to as "Ours" in Tables 3, 4, 5) consistently outperforms the baseline approaches in terms of Balanced Efficacy (BE), Wanted Efficacy (WE), and Unwanted Efficacy (UE), demonstrating a strong ability to unlearn unwanted instances, while preserving the performance on the wanted set. The method achieves this without the extreme variability seen in some baselines. The forget set produced by ForSId-via-RBSC is of reasonable size (FS-Size). This complies with the intuition that an excessively small forget set may fail to effectively address the FORGET-SET IDENTIFICATIONproblem, while an overly large one can degrade model's performance.

Among the baselines, k-NN provides reasonable results but lacks precision in distinguishing unwanted instances (i.e., UE), leading to moderate efficacy scores across all datasets. TopK-U is relatively competitive, particularly when some overlap exists between wanted and unwanted sets, but it does not consistently match the balanced efficacy of ForSId-via-RBSC. TopK-D struggles with unwanted efficacy, excelling in retaining wanted instances but failing to fully remove unwanted ones, leading to an imbalance that undermines its effectiveness overall. ImpDisp performs well in certain cases but exhibits significant variation, particularly in settings where the wanted and unwanted sets overlap substantially, making it less reliable across different scenarios.

In the ONE VS OTHERS setting (Table 3), where the distinction between wanted and unwanted sets is clear, and the unwanted set is well defined (i.e., sampled from only one class), our ForSId-via-RBSC achieves near-optimal values for BE, UE, and WE, proving that it effectively unlearns the targeted samples while maintaining correct predictions on the wanted set. TopK-U and k-NN perform adequately but struggle to maintain high BE.

TopK-D shows evident limitations, as its methodology of strictly differentiating wanted from unwanted instances does not generalize well in this setting. ImpDisp delivers inconsistent results across different datasets, reinforcing that while impact-based approaches can be useful, they are not universally effective.

When moving to the One vs all setting (Table 4), which introduces more complexity by allowing samples of the same class to belong to both wanted and unwanted sets, our ForSId-via-RBSC maintains superior efficacy scores but, as expected, with a slightly reduced margin compared to the (previous) clear separation setting. This highlights its robustness in handling uncertainty. TopK-U improves in this setting, benefiting from its ability to account for influence without rigid constraints. k-NN remains relatively stable but does not improve significantly. TopK-D continues to struggle due to its inability to handle cases where influence is not strictly binary. While still variable, ImpDisp shows some improvement, particularly in datasets where unwanted and wanted samples exhibit structured similarities (i.e., CIFAR's ones).

In the Many vs many setting (Table 5), where the wanted and unwanted sets are drawn from different sets of classes with increased variance, our ForSId-via-RBSC again leads in BE, UE, and WE. However, the accuracy loss (Acc. Δ) is higher than in previous settings, suggesting that the task difficulty introduces some trade-offs. However, it is important to note that accuracy alone does not give the full picture of the experiments. The sharp drop in accuracy in Table 5, which represents `many vs. many` experiments, is inherent within the problem's formulation. Since we are formulating multiple classes to be "Unwanted", a drop in accuracy is not only expected, but also somewhat desirable as the output of the classes represented in the unwanted set will be changed. Critically, our algorithm maintains near-perfect Wanted Efficacy, preserving the outputs of the sampled indicated as "Wanted" more than 94% of the times on CIFAR-10, and close to 99% on CIFAR-10 and CIFAR-100, as reported in Table 5. We also remind that `many vs many` is an edge case with maximum confusion among samples.

To further corroborate this point, note that the performance of the baselines drops consistently in this scenario. TopK-U and k-NN suffer the most, due to the higher complexity of separating the sets, while TopK-D, although performing slightly better than in the previous setting, remains suboptimal due to its inherent imbalance. ImpDisp particularly struggles, as its reliance on impact differences does not generalize well to highly disjoint distributions, reinforcing the importance of using more structured methodologies.

Looking across datasets, the trends remain consistent. The CIFAR-10, CIFAR-20, and CIFAR-100 image datasets exhibit similar behavior, where our ForSId-via-RBSC retains its superiority across efficacy measures, though some baselines achieve closer results in lower-complexity tasks. The Wine Quality dataset is tabular and smaller in scale, thus it exhibits less pronounced differences across methods, but ForSId-via-RBSC still maintains the highest efficacy scores, showing that its effectiveness is not limited to image data.

An important remark, valid for all the experimental settings, is that the effectiveness of our method is independent on the size of the identified forget set. In fact, as shown in Tables 3, 4, 5, the models exhibit minimal accuracy change (Acc. Δ) for CIFAR-10, CIFAR-20 and Wine, indicating that utility is well-maintained, while achieving high unlearning efficacy (UE), meaning that the unwanted set is effectively forgotten. The tables show that our method is among the methods that identify the smallest forget set (aside from a few cases), despite having the best overall performance on the selected metrics. That means that other

baselines need to remove at least as many samples, but still get outperformed (See Appendix 8.2 for an in-depth discussion). Determining the ideal or minimal size of a forget set for effective unlearning is a challenging theoretical problem, closely related to fundamental questions in learning theory, such as identifying the minimum number of samples needed to approximate a decision boundary (e.g., PAC learning (Valiant, 1984) and VC dimension (Vapnik, 1968)). Intuitively, the maximum forget set is the set that does not include the samples that approximate the decision boundary (with the exception of those that define the unwanted behaviour). Accordingly, the forget set could be very large indeed, especially if the training set is large and the decision boundary is relatively simple. While we believe this is an interesting and valuable direction for future work, a rigorous analysis is beyond the scope of our current paper. Moreover, adding a maximum amount of samples to be removed in the formulation of our problem is straightforward, but beyond the scope of this initial exploration.

Lastly, the analysis of the running time should be considered carefully. While ForSId-via-RBSC does have a higher running time than some baselines, this is part of bigger tradeoff. Many cases show that methods with lower running times are really ineffective, as they often remove too few samples or fail to achieve a meaningful separation between wanted and unwanted instances. The tradeoff between efficiency and effectiveness is crucial, and ForSId-via-RBSC 's performance justifies its computational cost. The observation that running time alone does not determine a method's viability is reinforced by the fact that some of the fastest baselines, such as TopK-D, deliver highly imbalanced results and, therefore, they are unpractical despite their lower computational cost.

Overall, the results indicate that the proposed ForSId-via-RBSC achieves a strong balance between efficacy, a reasonable forget set size, and an appropriate computational efficiency. It avoids the pitfalls of excessively large and excessively small forget sets, removing unwanted instances without disrupting the overall model performance. While some baselines provide useful comparative insights, none achieve the same consistency across different experimental settings and datasets. Further details on the analysis of the baseline parameter k and our parameter T are reported in Appendix A.

7 Conclusion

In this paper, we advance the state of the art in Machine Unlearning (MU) by introducing the novel Forget-Set Identification (ForSId) problem: given a trained machine-learning model, and user's unwanted and wanted evidence, find a subset of the training set utilized upstream to train the model (i.e., the "forget set") so that retraining the model on the set resulting from the removal of the forget set from the original training set makes the model unlearn the unwanted evidence and keep the wanted evidence. We formalize the ForSId problem, prove its NP-hard ness, and design an algorithm for it based on a connection to Red-Blue Set Cover. Extensive experiments attest high performance of the proposed algorithm. Notably, for future work, researching more solid parameter selection techniques, like bisection, could further improve the results of our algorithm. By defining and studying the ForSId problem, we fill a crucial gap in the MU landscape, allowing for extending the applicability of MU in all those contexts where providing a forget set as input to MU methods is not possible or overly difficult.

In the future, we plan to devise algorithms that tackle FORSID more directly (without passing through proxy problems), deal with approximation properties and algorithms for the proxy RED-BLUE FORSID problem, consider different forms of (un)wanted behavior, investigate ad-hoc methodologies for setting the parameters of our method(s), and experimentally test FORSID in additional scenarios and settings. Specifically, although ILP solvers are powerful, they have practical limitations in terms of scalability and computational efficiency. To address these constraints, we plan to investigate data engineering techniques such as instance selection (Cunha et al., 2023; Olvera-López et al., 2010) or genetic algorithms (Katoch et al., 2021), which can help us identify samples more efficiently.

Detailed results

In this appendix, we corroborate and integrate our results with more in-depth analysis. Specifically, in Sect. 8.1, we present the same tables of Sect. 6, enriched with standard deviations, previously omitted for brevity. In Sect. 8.2, we study the performance trends of FORSID and the baselines as a function of parameters T and k, respectively.

Standard deviation

Here we show Tables 6,7,8, that list the same values of Tables 3,4,5 (shown in Sect. 6 but enriched with standard deviation. The values shown in the following tables are the mean and standard deviations of 10 runs, as described in Sect. 5. The standard deviations reported in these tables are generally small compared to their respective means, indicating that the results are stable across the 10 runs.

This proves that the observed performance patterns are not artifacts of random variation but rather reflect genuine properties of the methods, reinforcing the reliability of the results presented in Sect. 6.

Across all scenarios, we observe that the standard deviations tend to be larger on the Wine dataset compared to the CIFAR datasets. This is an expected outcome given the characteristics of Wine: it is substantially smaller in both sample size and feature dimensionality, which makes its evaluation metrics more sensitive to minor fluctuations in predictions. Furthermore, the Wine dataset suffers from severe class imbalance, where some classes contain only a handful of samples while others constitute the majority of the dataset (Cortez et al., 2009). As a result, any small change in the predicted labels can lead to considerable variations in evaluation metrics such as accuracy or F1-score, amplifying the observed standard deviations.

On the other hand, the CIFAR datasets, which are much larger and more balanced in class distributions, exhibit consistently low standard deviations. This is particularly evident for the FORSID method, which maintains remarkably stable performance across all repetitions, regardless of the class or scenario under evaluation. This consistency reinforces its capacity to outperform baseline methods reliably, providing further evidence of its robustness and effectiveness in practical settings. Overall, these additional results support the validity of the findings reported in Sect. 6.

Table 6 Comparison of ForSId-via-RBSC (Ours) with baselines methods

		BE ↑	UE ↑	WE ↑	Acc. Δ↓	H ↑	FS-Size	RunT (s) ↓
CIFAR-10	k-NN	0.605 ± 0.071	0.554 ± 0.100	0.657 ± 0.070	0.191 ± 0.036	0.639 ± 0.064	36814.818 ± 2400.515	58.122 ± 11.309
	TopK-U	0.639 ± 0.065	0.345 ± 0.121	0.932 ± 0.011	0.018 ± 0.004	0.585 ± 0.126	10205.091 ± 462.544	50.029 ± 0.781
	TopK-D	0.525 ± 0.010	0.055 ± 0.019	0.996 ± 0.003	0.001 ± 0.004	0.148 ± 0.045	613.182 ± 136.549	25.622 ± 0.872
	ImpDisp	0.655 ± 0.094	0.522 ± 0.150	0.788 ± 0.050	0.122 ± 0.030	0.676 ± 0.100	24278.250 ± 1736.286	2658.045 ± 18.428
	Ours	0.982 ± 0.008	0.999 ± 0.011	0.965 ± 0.009	0.077 ± 0.017	0.952 ± 0.012	19266.000 ± 743.27	1094.689 ± 15.321
CIFAR-20	k-NN	0.767 ± 0.053	0.608 ± 0.106	0.927 ± 0.007	0.011 ± 0.004	0.795 ± 0.063	7676.500 ± 553.122	149.123 ± 112.643
	TopK-U	0.843 ± 0.039	0.798 ± 0.079	0.889 ± 0.020	0.014 ± 0.006	0.878 ± 0.034	8988.100 ± 671.028	40.479 ± 8.478
	TopK-D	0.661 ± 0.021	0.322 ± 0.042	1.000 ± 0.001	0.006 ± 0.003	0.584 ± 0.046	1065.778 ± 40.335	31.510 ± 6.247
	ImpDisp	0.790 ± 0.060	0.948 ± 0.052	0.631 ± 0.086	0.102 ± 0.022	0.766 ± 0.062	25296.900 ± 3539.419	3223.776 ± 530.344
	Ours	0.986 ± 0.042	0.993 ± 0.023	0.979 ± 0.062	0.011 ± 0.024	0.983 ± 0.043	5247.000 ± 1208.212	1226.710 ± 112.642
CIFAR-100	k-NN	0.774 ± 0.099	0.626 ± 0.188	0.921 ± 0.012	0.034 ± 0.017	0.782 ± 0.121	8308.800 ± 653.706	20.272 ± 8.630
	TopK-U	0.902 ± 0.101	0.867 ± 0.193	0.936 ± 0.027	0.014 ± 0.004	0.913 ± 0.093	5098.500 ± 621.340	1.918 ± 0.450
	TopK-D	0.835 ± 0.074	0.669 ± 0.147	1.000 ± 0.000	0.007 ± 0.007	0.846 ± 0.087	672.400 ± 126.945	3.714 ± 0.753
	ImpDisp	0.799 ± 0.054	0.968 ± 0.091	0.630 ± 0.028	0.173 ± 0.017	0.742 ± 0.036	24618.100 ± 1731.776	322.440 ± 56.844
	Ours	0.968 ± 0.068	0.943 ± 0.139	0.993 ± 0.008	0.152 ± 0.021	0.873 ± 0.043	17023.667 ± 836.755	187.575 ± 50.513
Wine	k-NN	0.855 ± 0.128	0.880 ± 0.199	0.830 ± 0.064	0.038 ± 0.019	0.871 ± 0.106	2508.200 ± 1543.119	0.179 ± 0.184
	TopK-U	0.768 ± 0.157	0.800 ± 0.355	0.735 ± 0.089	0.039 ± 0.008	0.764 ± 0.181	2186.250 ± 218.494	1.326 ± 1.384
	TopK-D	0.744 ± 0.310	0.708 ± 0.450	0.780 ± 0.175	-0.015 ± 0.037	0.735 ± 0.405	115.500 ± 30.359	0.239 ± 0.071
	ImpDisp	0.775 ± 0.310	0.761 ± 0.450	0.790 ± 0.175	0.021 ± 0.037	0.763 ± 0.405	2310.600 ± 30.359	19.750 ± 2.147
	Ours	0.920 ± 0.030	1.000 ± 0.000	0.840 ± 0.020	0.028 ± 0.017	0.924 ± 0.010	1795.000 ± 439.820	9.848 ± 0.324

Benchmarking measures as the mean of 10 runs. Bold values are the best overall; underlined ones are second best. Efficacies (BE, UE, WE) are the most important measuresOne vs others

Table 7 Comparison of ForSId-via-RBSC (Ours) with baselines methods

		BE ↑	UE ↑	WE ↑	Acc. Δ↓	H ↑	FS-Size	RunT (s) ↓
CIFAR-10	k-NN	0.593 ± 0.065	0.560 ± 0.102	0.627 ± 0.061	0.190 ± 0.037	0.632 ± 0.060	36660.100 ± 2471.182	54.281 ± 6.424
	TopK-U	0.626 ± 0.058	0.349 ± 0.128	0.903 ± 0.019	0.018 ± 0.004	0.581 ± 0.130	10137.400 ± 441.964	50.680 ± 0.790
	TopK-D	0.503 ± 0.002	0.007 ± 0.005	0.999 ± 0.001	-0.001 ± 0.004	0.021 ± 0.013	38.000 ± 14.252	6.135 ± 0.200
	ImpDisp	0.661 ± 0.117	0.562 ± 0.225	0.760 ± 0.037	0.122 ± 0.023	0.662 ± 0.198	24621.300 ± 2414.427	2663.667 ± 58.910
	Ours	0.802 ± 0.011	0.695 ± 0.032	0.910 ± 0.012	0.033 ± 0.014	0.837 ± 0.078	10297.000 ± 4405.850	1281.942301 ± 32.488
CIFAR-20	k-NN	0.760 ± 0.049	0.626 ± 0.104	0.895 ± 0.012	0.007 ± 0.005	0.799 ± 0.059	9828.250 ± 644.165	48.670 ± 13.028
	TopK-U	0.823 ± 0.038	0.820 ± 0.079	0.826 ± 0.015	0.010 ± 0.005	0.867 ± 0.030	11907.875 ± 971.407	36.859 ± 1.116
	TopK-D	0.516 ± 0.003	0.032 ± 0.006	0.999 ± 0.001	-0.004 ± 0.003	0.090 ± 0.015	93.875 ± 11.777	5.903 ± 0.180
	ImpDisp	0.793 ± 0.024	0.948 ± 0.033	0.638 ± 0.034	0.074 ± 0.009	0.788 ± 0.023	23874.500 ± 1934.396	2664.712 ± 127.513
	Ours	0.850 ± 0.008	0.864 ± 0.020	0.835 ± 0.004	0.022 ± 0.004	0.880 ± 0.001	6858.000 ± 125.865	1237.051 ± 8.424
CIFAR-100	k-NN	0.800 ± 0.064	0.669 ± 0.130	0.931 ± 0.012	0.026 ± 0.006	0.823 ± 0.067	6963.625 ± 465.010	22.702 ± 10.069
	TopK-U	0.926 ± 0.034	0.963 ± 0.062	0.889 ± 0.027	0.020 ± 0.024	0.936 ± 0.028	5156.375 ± 622.996	1.890 ± 0.374
	TopK-D	0.676 ± 0.188	0.355 ± 0.380	0.996 ± 0.007	0.001 ± 0.004	0.441 ± 0.465	524.250 ± 217.462	4.170 ± 1.139
	ImpDisp	0.829 ± 0.034	1.000 ± 0.000	0.657 ± 0.069	0.177 ± 0.019	0.757 ± 0.035	24636.125 ± 1699.436	330.794 ± 72.977
	Ours	0.988 ± 0.008	1.000 ± 0.000	0.977 ± 0.015	0.187 ± 0.004	0.858 ± 0.006	20417.667 ± 260.316	112.443 ± 6.469
Wine	k-NN	0.797 ± 0.093	0.820 ± 0.157	0.775 ± 0.078	0.033 ± 0.017	0.832 ± 0.075	2368.500 ± 946.219	0.216 ± 0.095
	TopK-U	0.661 ± 0.167	0.728 ± 0.278	0.595 ± 0.058	0.024 ± 0.052	0.719 ± 0.140	3875.000 ± 116.167	3.324 ± 1.587
	TopK-D	0.587 ± 0.218	0.372 ± 0.403	0.802 ± 0.051	0.039 ± 0.012	0.488 ± 0.309	88.750 ± 92.662	0.374 ± 0.070
	ImpDisp	0.661 ± 0.105	0.615 ± 0.229	0.708 ± 0.028	0.001 ± 0.016	0.689 ± 0.129	2457.750 ± 117.452	25.294 ± 0.459
	Ours	0.744 ± 0.187	0.765 ± 0.359	0.723 ± 0.062	-0.029 ± 0.024	0.778 ± 0.226	2319.000 ± 194.504	11.209 ± 0.722

Benchmarking measures as the mean of 10 runs. Bold values are the best overall; underlined ones are second best. Efficacies (BE, UE, WE) are the most important measuresOne vs all

Table 8 Comparison of ForSId-via-RBSC (Ours) with baselines methods

		BE ↑	UE ↑	WE ↑	Acc. Δ↓	H ↑	FS-Size	RunT (s) ↓
CIFAR-10	$k-NN$	0.552 ± 0.089	0.476 ± 0.072	0.628 ± 0.125	0.216 ± 0.034	0.586 ± 0.070	39190.600 ± 2164.686	121.306 ± 92.484
	$TopK-U$	0.579 ± 0.034	0.327 ± 0.028	0.831 ± 0.042	0.072 ± 0.010	0.559 ± 0.034	27916.400 ± 794.768	75.401 ± 13.846
	$TopK-D$	0.543 ± 0.013	0.093 ± 0.025	0.994 ± 0.006	0.017 ± 0.006	0.233 ± 0.055	6436.600 ± 816.194	103.775 ± 19.558
	$ImpDisp$	0.540 ± 0.025	0.265 ± 0.052	0.814 ± 0.065	0.103 ± 0.040	0.481 ± 0.044	23792.000 ± 2973.867	3276.857 ± 825.794
	Ours	0.869 ± 0.068	0.793 ± 0.092	0.944 ± 0.053	0.262 ± 0.037	0.776 ± 0.025	30727.800 ± 1127.532	1065.329 ± 10.010
CIFAR-20	$k-NN$	0.752 ± 0.011	0.534 ± 0.016	0.970 ± 0.007	0.001 ± 0.006	0.768 ± 0.011	3838.429 ± 59.997	46.738 ± 4.973
	$TopK-U$	0.733 ± 0.013	0.660 ± 0.012	0.806 ± 0.019	0.061 ± 0.004	0.769 ± 0.010	27496.143 ± 466.265	25.109 ± 0.930
	$TopK-D$	0.666 ± 0.008	0.333 ± 0.015	1.000 ± 0.001	0.009 ± 0.005	0.597 ± 0.015	10531.600 ± 176.618	25.686 ± 0.891
	$ImpDisp$	0.539 ± 0.039	0.370 ± 0.039	0.709 ± 0.055	0.065 ± 0.012	0.567 ± 0.038	23653.429 ± 1909.292	2717.737 ± 54.341
	Ours	0.945 ± 0.021	0.895 ± 0.042	0.996 ± 0.003	0.138 ± 0.010	0.849 ± 0.009	23344.200 ± 758.332	1314.937 ± 45.560
CIFAR-100	$k-NN$	0.722 ± 0.013	0.485 ± 0.022	0.958 ± 0.004	0.012 ± 0.003	0.727 ± 0.019	6152.000 ± 111.903	47.285 ± 4.797
	$TopK-U$	0.800 ± 0.005	0.787 ± 0.006	0.813 ± 0.017	0.189 ± 0.013	0.755 ± 0.011	28024.000 ± 319.839	13.336 ± 0.074
	$TopK-D$	0.709 ± 0.011	0.421 ± 0.020	0.997 ± 0.002	0.104 ± 0.007	0.653 ± 0.012	15558.000 ± 77.216	31.007 ± 2.626
	$ImpDisp$	0.467 ± 0.013	0.380 ± 0.003	0.553 ± 0.025	0.188 ± 0.008	0.508 ± 0.007	25987.000 ± 373.101	2724.077 ± 64.654
	Ours	0.998 ± 0.068	1.000 ± 0.092	0.997 ± 0.053	0.256 ± 0.037	0.795 ± 0.025	24534.000 ± 1127.532	1756.515 ± 93.538
Wine	$k-NN$	0.857 ± 0.071	1.000 ± 0.007	0.714 ± 0.148	0.032 ± 0.042	0.866 ± 0.081	5196.000 ± 255.766	1.728 ± 2.932
	$TopK-U$	0.921 ± 0.175	1.000 ± 0.304	0.842 ± 0.046	0.024 ± 0.049	0.927 ± 0.163	5194.000 ± 188.794	59.013 ± 0.066
	$TopK-D$	0.789± 0.157	0.697 ± 0.332	0.882 ± 0.017	0.073 ± 0.015	0.804 ± 0.298	1853.000 ± 263.272	23.323 ± 0.119
	$ImpDisp$	0.752 ± 0.157	0.658 ± 0.332	0.845 ± 0.017	0.014 ± 0.015	0.804 ± 0.298	2729.000 ± 263.272	262.115 ± 0.119
	Ours	0.801 ± 0.165	0.752 ± 0.338	0.851 ± 0.009	-0.003 ± 0.040	0.857 ± 0.104	2638.000 ± 9.238	124.070± 7.310

Benchmarking measures as the mean of 10 runs. Bold values are the best overall; underlined ones are second best. Efficacies (BE, UE, WE) are the most important measuresMANY VS MANY

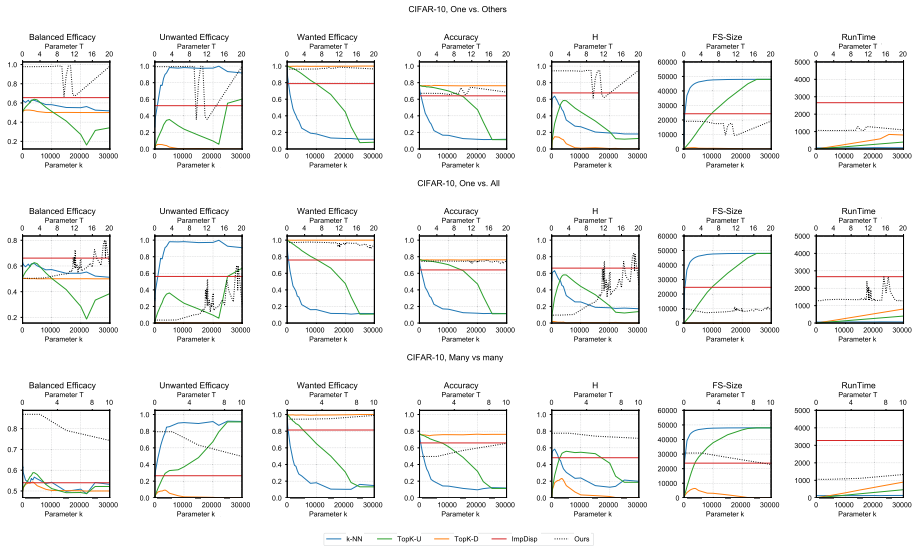

Fig. 2 Comparison of ForSId-via-RBSC (Ours) with baseline methods on CIFAR-10 across three experimental settings: One vs others, One vs all, and Many vs many, as a function of parameters k and t. The x-axis below represents parameter K, while the x-axis above represents parameter T for Forget-Set Identification

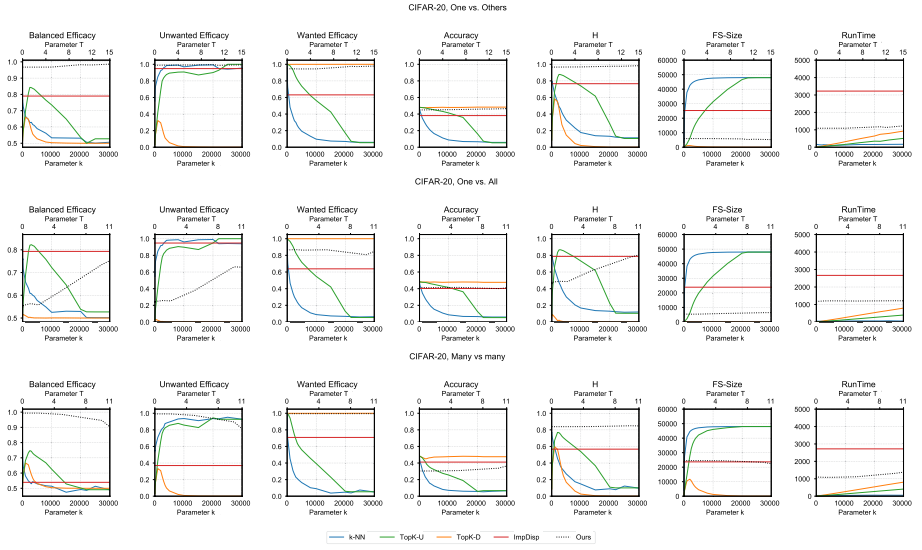

Fig. 3 Comparison of ForSId-via-RBSC (Ours) with baseline methods on CIFAR-20 across three experimental settings: One vs others, One vs all, and Many vs many, as a function of parameters k and t. The x-axis below represents parameter K, while the x-axis above represents parameter T for Forget-Set Identification

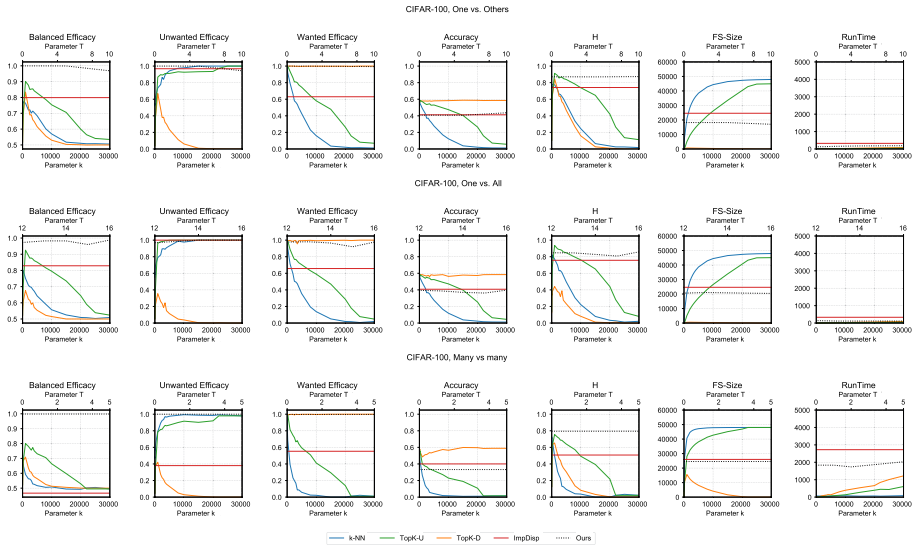

Fig. 4 Comparison of ForSId-via-RBSC (Ours) with baseline methods on CIFAR-100 across three experimental settings: One vs others, One vs all, and Many vs many, as a function of parameters k and t. The x-axis below represents parameter K, while the x-axis above represents parameter T for Forget-Set Identification

Parameter study and trends

In this Section, we show a parameter study on parameters T (for our method ForSId) and k (for all the baselines). We report results for all tested parameter values and analyze the behavior of ForSId and the baselines as these parameters approach the total number of samples in the dataset.

Figure 2 shows the the trends exhibited by each baseline and ForSId when parameters K and T change, for each scenario (from top to bottom). The figure clearly conveys the patterns of each method's behavior under these parameter changes. k-NN and TopK-U have very similar behavior, as they are very similar methods (refer to Sect. 5: when parameter K increases, both tend to become extremely disruptive, increasing their Unwanted Efficacy but at the same time plummeting the Wanted Efficacy, as they are destroying the utility of the model and flattening the predictions as a result. This is also evident from the stark drop in accuracy they cause on all three scenarios and the ever-increasing size of the Forget Set they identify. This is expected: both k-NN and TopK-U strictly increase the size of the Forget Set when K increases.

On the other hand, TopK-D exhibits the opposite behavior: since increasing the parameter K builds a bigger set to remove from the initial set of identified samples, when K increases too much, the Forget Set identified by TopK-D collapses to 0 (refer to FS-Size across all three scenarios). As a result, applying this method is basically like doing nothing: the Wanted Efficacy is extremely high (as no wanted samples are changing prediction) but the Unwanted Efficacy is 0 (as no unwanted samples are changing prediction either), effectively making the method useless.

ImpDisp does not depend on any parameter and is represented by a straight line.

Finally, ForSId is consistently the best performer overall. While each of the baselines, for the aforementioned reason, excel in one area while collapsing on all others, ForSId demonstrates balanced and robust performance across all evaluated metrics and scenarios. This is well captured by the metric H, an harmonic mean of all the others (refer to Sect. 5), where ForSId outperforms all others, often by a large margin. The change of parameter T does have an impact on ForSId, but it's often negligible with respect to other baselines, which are by far more impacted by the change of parameter K. However, a parameter tuning on ForSId can yield better results.

Moreover, from these trends, it is clear how ForSId is also among the methods that find the smallest Forget Set (excluding TopK-D, that as we explained collapses to 0), while still selecting the right samples to overperform the other baselines across all metrics.

For the sake of completeness, Figs. 3 and 4 illustrate the same trends for CIFAR-20 and CIFAR-100, respectively. As the observed patterns are consistent with those already discussed, we omit further detailed commentary here. However, both images show how ForSId outperforms the baseline while consistently finding the smallest Forget Sets.

Acknowledgements The numerical simulations have been realized on the HPC cluster of the Department of Information Engineering, Computer Science and Mathematics (DISIM) at the University of L'Aquila. The work is partially funded by the European Union - NextGenerationEU under the Italian Ministry of University and Research (MUR) National Innovation Ecosystem grant ECS00000041 - VITALITY - CUP E13C22001060006, by National Recovery and Resilience Plan (Piano Nazionale di Ripresa e Resilienza, PNRR) - Project: "SoBigData.it - Strengthening the Italian RI for Social Mining and Big Data Analytics" - Prot. IR0000013 - Avviso n. 3264 del 28/12/2021, and by the "ICSC – Centro Nazionale di Ricerca in High Performance Computing, Big Data and Quantum Computing."

Author contributions Conceptualization: A.D., F.G., G.S; Methodology: A.D., F.G., G.S.; Investigation:A.D.; Software:A.D.,G.S.; Formal analysis and investigation: A.D., F.G., G.S.; Writing - original draft preparation: A.D., F.G., G.S.; Writing - review and editing: A.D., F.G., G.S.; Visualization:A.D.; Supervision: F.G., G.S.;

Funding Open access funding provided by Luiss University within the CRUI-CARE Agreement.

Data availability No datasets were generated or analysed during the current study.

Declarations

Conflict of interest The authors declare no Conflict of interest.

Declaration of generative AI and AI-assisted technologies in the writing process During the preparation of this work, the authors used AI-assisted technologies to correct typos and grammatical mistakes. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

References

Abidha, V. P., & Ashok, P. (2024). Red blue set cover problem on axis-parallel hyperplanes and other objects. *Information Processing Letters, 186*, Article 106485.

Ashok, P., Kolay, S., & Saurabh, S. (2017). Multivariate complexity analysis of geometric red blue set cover. *Algorithmica, 79*(3), 667–697.

Birhane, A., Prabhu, V. U., & Kahembwe, E. (2021). Multimodal datasets: Misogyny, pornography, and malignant stereotypes. CoRR **abs/2110.01963**.

Bonato, J., Cotogni, M., & Sabetta, L. (2024). Is retain set all you need in machine unlearning? restoring performance of unlearned models with out-of-distribution images. In: Proc. of European Conf. on Computer Vision (ECCV), 1–19.

Cadamuro, G., Gilad-Bachrach, R., & Zhu, J. (2016). Debugging machine learning models. https://api.semanticscholar.org/CorpusID:14629198.

Cao, Y., & Yang, J. (2015). Towards making systems forget with machine unlearning. In: Proc. of IEEE Symposium on Security and Privacy (S &P), 463–480.

Caprara, A., Toth, P., & Fischetti, M. (2000). Algorithms for the set covering problem. *Annals of Operations Research, 98*(1), 353–371. https://doi.org/10.1023/A:1019225027893

Carr, R.D., Doddi, S., Konjevod, G., & Marathe, M.V. (2000). On the red-blue set cover problem. In: Proc. of ACM-SIAM Symp. on Discrete Algorithms (SODA), 345–353.

Chan, T. M., & Hu, N. (2015). Geometric red-blue set cover for unit squares and related problems. *Computational Geometry: Theory and Applications, 48*(5), 380–385.

Chen, R., Yang, J., Xiong, H., Bai, J., Hu, T., Hao, J., Feng, Y., Zhou, J. T., Wu, J., & Liu, Z. (2023). Fast model debias with machine unlearning. In: Proc. of Conf. on Advances in Neural Information Processing Systems (NeurIPS).

Chlamtác, E., Makarychev, Y., & Vakilian, A. (2023). Approximating red-blue set cover and minimum monotone satisfying assignment. In: Proc. of Int. Conf. on Approximation Algorithms for Combinatorial Optimization Problems and Int. Conf. on Randomization and Computation (APPROX/RANDOM), 11–11119.

Chundawat, V.S., Tarun, A.K., Mandal, M., & Kankanhalli, M.S. (2023). Can bad teaching induce forgetting? Unlearning in deep networks using an incompetent teacher. In: Proc. of AAAI Conf. on Artificial Intelligence (AAAI), 7210–7217.

Chundawat, V. S., Tarun, A. K., Mandal, M., & Kankanhalli, M. (2023). Zero-shot machine unlearning. *IEEE Transactions on Information Forensics and Security, 18*, 2345–2354. https://doi.org/10.1109/TIFS.2023.3265506

Cortez, P., Cerdeira, A., Almeida, F., Matos, T., & Reis, J. (2009). *Wine Quality. UCI Machine Learning Repository.* https://doi.org/10.24432/C56S3T

Cunha, W., Viegas, F., França, C., Rosa, T., Rocha, L., & Gonçalves, M. A. (2023). A comparative survey of instance selection methods applied to non-neural and transformer-based text classification. *ACM Computing Surveys*. https://doi.org/10.1145/3582000

D'Angelo, A., Savelli, C., Tagliente, G., Giobergia, F., Baralis, E., & Stilo, G. (2025). How to make reproducible research in machine unlearning with ERASURE.In: Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence (IJCAI-25). International Joint Conferences on Artificial Intelligence Organization, 11025–11029. https://doi.org/10.24963/ijcai.2025/1255

Dang, Q.-V. (2021). Right to be forgotten in the age of machine learning. In: Advances in Digital Science, 403–411.

Elkin, M., & Peleg, D. (2007). The hardness of approximating spanner problems. *Theory of Computing Systems (TOCS), 41*(4), 691–729.

European Parliament and Council of the European Union: Regulation of the European Parliament and of the Council laying down harmonised rules on artificial intelligence (Artificial Intelligence Act) and amending certain Union legislative acts. COM/2021/206 final (2021). https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A52021PC0206.

Fan, C., Liu, J., Hero, A.O., & Liu, S. (2024). Challenging forgets: Unveiling the worst-case forget sets in machine unlearning. In: Proc. of European Conf. on Computer Vision (ECCV), 278–297.

Foster, J., Schoepf, S., & Brintrup, A. (2024). Fast machine unlearning without retraining through selective synaptic dampening. In: Proc. of AAAI Conf. on Artificial Intelligence (AAAI), 12043–12051.

Gandikota, R., Materzynska, J., Fiotto-Kaufman, J., & Bau, D. (2023). Erasing concepts from diffusion models. In: Proc. of IEEE Int. Conf. on Computer Vision (ICCV), 2426–2436.

Ginart, A., Guan, M.Y., Valiant, G., & Zou, J. (2019). Making AI forget you: Data deletion in machine learning. In: Proc. of Conf. on Advances in Neural Information Processing Systems (NeurIPS), 3513–3526.

Golatkar, A., Achille, A., & Soatto, S. (2020). Eternal sunshine of the spotless net: Selective forgetting in deep networks. In: Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 9301–9309.
Katoch, S., Chauhan, S. S., & Kumar, V. (2021). A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications, 80*, 8091–8126. https://doi.org/10.1007/s11042-020-10139-6
Kellerer, H., Pferschy, U., & Pisinger, D. (2004). Knapsack Problems. Springer.
Koh, P.W., & Liang, P. (2017). Understanding black-box predictions via influence functions. In: Proc. of Int. Conf. on Machine Learning (ICML), 1885–1894.
Krizhevsky, A., & Hinton, G. (2010). Convolutional deep belief networks on CIFAR-10. Unpublished manuscript, 40(7), 1–9.
Kung, S. Y. (2014). *Kernel Methods and Machine Learning*. Cambridge University Press.
Kuwana, Y., Goto, Y., Shibata, T., & Irie, G. (2024). Black-box forgetting. In: Proc. of Conf. on Advances in Neural Information Processing Systems (NeurIPS).
Kwak, C., Lee, J., Park, K., & Lee, H. (2017). Let machines unlearn—machine unlearning and the right to be forgotten. In: Proc. of Americas Conf. on Information Systems (AMCIS).
Lev, O., & Wilson, A. (2024). Faster Machine Unlearning via Natural Gradient Descent. arXiv:2407.08169.
Liu, Y., Sun, C., Wu, Y., & Zhou, A. (2023). Unlearning with Fisher Masking. arXiv:2310.05331.
Ly, A., Marsman, M., Verhagen, J., Grasman, R. P. P. P., & Wagenmakers, E.-J. (2017). A tutorial on fisher information. *Journal of Mathematical Psychology, 80*, 40–55.
Madireddy, R. R., & Mudgal, A. (2023). A constant-factor approximation algorithm for red-blue set cover with unit disks. *Algorithmica, 85*(1), 100–132.
Mantelero, A. (2013). The EU proposal for a general data protection regulation and the roots of the 'right to be forgotten'. *Computer Law & Security Review, 29*(3), 229–235.
Newatia, A., Cooper, M., & Krishnan, R. (2024). Unlearning tabular data without a "forget set". In: Proc. of the Third Table Representation Learning Workshop @NeurIPS 2024.
Olvera-López, J. A., Carrasco-Ochoa, J. A., Martínez-Trinidad, J. F., & Kittler, J. (2010). A review of instance selection methods. *Artificial Intelligence Review, 34*(2), 133–143. https://doi.org/10.1007/s10462-010-9165-y
Shibata, T., Irie, G., Ikami, D., & Mitsuzumi, Y. (2021). Learning with selective forgetting. In: Proc. of Int. Joint Conf. on Artificial Intelligence (IJCAI), 989–996.
Tarun, A. K., Chundawat, V. S., Mandal, M., & Kankanhalli, M. S. (2024). Fast yet effective machine unlearning. *IEEE Transactions on Neural Networks and Learning Systems (TNNLS), 35*(9), 13046–13055.
Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM, 27*(11), 1134–1142.
Vapnik, V. (1968). On the uniform convergence of relative frequencies of events to their probabilities. *Doklady Akademii Nauk USSR, 181*, 781–787.
Wachter, S., Mittelstadt, B., & Floridi, L. (2017). Why a right to explanation of automated decision-making does not exist in the general data protection regulation. *International Data Privacy Law, 7*(2), 76–99.
Wang, S., Zhu, Y., Liu, H., Zheng, Z., Chen, C., & Li, J. (2025). Knowledge editing for large language models: A survey. *ACM Computing Surveys (CSUR), 57*(3), 59–15937.
Xu, H., Zhu, T., Zhang, L., Zhou, W., & Yu, P. S. (2024). Machine unlearning: A survey. *ACM Computing Surveys (CSUR), 56*(1), 9–1936.
Ye, J., Fu, Y., Song, J., Yang, X., Liu, S., Jin, X., Song, M., & Wang, X. (2022). Learning with recoverable forgetting. In: Proc. of European Conf. on Computer Vision (ECCV), 87–103.
Yeh, C., Kim, J.S., Yen, I.E., & Ravikumar, P. (2018). Representer point selection for explaining deep neural networks. In: Proc. of Conf. on Advances in Neural Information Processing Systems (NeurIPS), 9311–9321.
Yu, C., Jeoung, S., Kasi, A., Yu, P., & Ji, H. (2023). Unlearning bias in language models by partitioning gradients. In: Findings of the Association for Computational Linguistics: ACL, 6032–6048.
Zhao, K., Kurmanji, M., Barbulescu, G., Triantafillou, E., & Triantafillou, P. (2024). What makes unlearning hard and what to do about it. In: Proc. of Conf. on Advances in Neural Information Processing Systems (NeurIPS).

Authors and Affiliations

Andrea D'Angelo[1] · Francesco Gullo[1] · Giovanni Stilo[2,3]

✉ Giovanni Stilo
gstilo@luiss.it

Andrea D'Angelo
andrea.dangelo6@graduate.univaq.it

Francesco Gullo
francesco.gullo@univaq.it

[1] Department of Computer Science and Engineering and Mathematics, University of L'Aquila, 67100 L'Aquila, Italy

[2] Department of AI, Data and Decision Sciences, Luiss Guido Carli, Viale Romania, 32, 00197 Rome, Italy

[3] School of Management, Luiss Business School, Via Nomentana, 216, 00162 Rome, Italy