Machine Unlearning of Features and Labels

Alexander Warnecke*, Lukas Pirch*, Christian Wressnegger† and Konrad Rieck*
*Technische Universität Braunschweig
† KASTEL Security Research Labs, Karlsruhe Institute of Technology (KIT)

***Abstract*—Removing information from a machine learning model is a non-trivial task that requires to partially revert the training process. This task is unavoidable when sensitive data, such as credit card numbers or passwords, accidentally enter the model and need to be removed afterwards. Recently, different concepts for machine unlearning have been proposed to address this problem. While these approaches are effective in removing individual data points, they do not scale to scenarios where larger groups of features and labels need to be reverted. In this paper, we propose the first method for unlearning features and labels. Our approach builds on the concept of influence functions and realizes unlearning through closed-form updates of model parameters. It enables to adapt the influence of training data on a learning model retrospectively, thereby correcting data leaks and privacy issues. For learning models with strongly convex loss functions, our method provides certified unlearning with theoretical guarantees. For models with non-convex losses, we empirically show that the unlearning of features and labels is effective and significantly faster than other strategies.**

I. Introduction

Machine learning has become an ubiquitous tool in analyzing personal data and developing data-driven services. Unfortunately, the underlying learning models can pose a privacy threat if they inadvertently capture sensitive information from the training data and later reveal it to users. For example, Carlini et al. [1] show that the Google text completion system contains credit card numbers from personal emails, which may be exposed to other users during auto-completion. In addition, privacy regulations, such as the European GDPR [2], enable users to request the removal of their personal data from learning models as part of the "right to be forgotten".

Deleting data from a learning model is a challenging task that requires selectively reverting the learning process. In the absence of specific methods, the only option is to retrain the model from scratch, which is costly and only possible if the original data is still available. As a remedy, Cao and Yang [3] and Bourtoule et al. [4] propose methods for *machine unlearning*. These methods partially reverse the learning process and are capable of deleting learned data points in retrospection. As a result, they enable to mitigate privacy leaks and comply with removal requests from users.

Information leaks, however, do not only manifest in isolated data points. When training on content of social media, sensitive data is often distributed across several data instances. For example, the leaked home address of a celebrity may be shared in thousands of posts, rendering the removal of affected data points inefficient. Similarly, when applying machine learning on emails, personal data in conversations, such as names, addresses, and phone numbers, can affect dozens of messages and form features in the learning model whose later removal requires substantial changes to the model's structure.

Existing approaches for unlearning [3–8] are inefficient in these cases, as they operate on data points only: First, a runtime improvement can hardly be obtained over retraining when the changes are not isolated and larger parts of the data need to be corrected. Second, removing multiple data points reduces the fidelity of the corrected model and thus is not a viable option in practical scenarios. Consequently, unlearning should not be limited to removing data points, but allow corrections at different granularity of the training data, such as fixing leaks in features and labels individually.

In this paper, we propose the first method for unlearning features and labels from a learning model. Our approach is inspired by the concept of *influence functions*, a technique from robust statistics [9], that allows for estimating the influence of data on learning models [10, 11]. By reformulating this influence estimation as a form of unlearning, we derive a versatile approach that maps changes of the training data in retrospection to closed-form updates of model parameters. These updates can be calculated efficiently, even if larger parts of the training data are affected, and enable correcting features and labels captured within the model. As a result, our method can remove privacy leaks and other unwanted content from a wide range of common learning models.

For models with strongly convex loss, such as logistic regression and support vector machines, we prove that our approach enables *certified unlearning*. That is, it provides theoretical guarantees on the removal of features and labels from the models. To obtain these guarantees, we build on the concepts of certified data removal [5, 7] and differential privacy [12, 13]. In particular, we measure the difference between models obtained using our approach and retraining on corrected data. By carefully introducing noise into the learning process, we derive an upper bound on this difference and thus realize provable unlearning in practice.

For models with non-convex loss functions, such as deep neural networks, similar guarantees cannot be realized. However, we empirically demonstrate that our approach is significantly faster in comparison to sharding [4, 6] and retraining while reaching a similar level of accuracy. Moreover, due to the compact updates, our approach requires only a fraction of the training data and hence is applicable when the original data is not available. We demonstrate the efficacy of our approach in case studies on unlearning (a) sensitive features in linear models, (b) unintended memorization in language models, and (c) label poisoning in computer vision.

Network and Distributed System Security (NDSS) Symposium 2023
28 February - 4 March 2023, San Diego, CA, USA
ISBN 1-891562-83-5
https://dx.doi.org/10.14722/ndss.2023.23xxx
www.ndss-symposium.org

Threat model. For our approach, we consider learning models trained on privacy-sensitive data and accessible to users through an interface, such as a text completion system, a learning chatbot, or a collaborative spam filter. As these models operate on sensitive data, there is a need to protect their users' privacy and to close leaks in their interfaces as soon as possible.

1) First, this need arises from privacy regulations, such as the European GDPR. According to the GDPR, European citizens can request their personal information to be removed from a service to protect their privacy, including derived data in learning models [2].
2) Second, unintended memorization of personal data, such as credit card numbers, may also demand mechanisms for its removal. For example, recent work shows that text completion systems allow adversaries to extract sensitive data through their interfaces [1, 15].
3) Third, data used for constructing a learning model may accidentally violate ethical standards and thus also require removal once these violations have been detected. For example, the chatbot "Tay" by Microsoft memorized and replicated anti-semitic and racist content [34].

In all these cases, service providers must mitigate the resulting threats and immediately correct the learning model to reduce potential harm to their users.

From retraining to unlearning. At a first glance, retraining from scratch may seem like the optimal strategy to fix issues in learning models. By correcting the training data directly, all of the above issues can be reliably resolved. However, retraining from scratch comes with disadvantages:

1) Depending on the size of the original data, retraining from scratch can be costly. The entire learning process needs to be reproduced even if only a few data instances, features, or labels need to be corrected.
2) Privacy regulations require that the purpose and duration of data storage are clearly defined and approved by the user. Therefore, it may not be possible to keep the original data indefinitely for retraining.
3) Finally, with an online learning system like a chatbot, training data is volatile and might not be fully available. Nevertheless, even with such systems, inappropriate memorization must be removed quickly.

As a result of this situation, different concepts for machine unlearning have been proposed in the last years [3–8]. We follow this line of work and introduce a new mechanism for removing features and labels from learning models.

Unlearning instances vs. features. In many learning-based systems, data points are directly linked to individuals. For example, in a face recognition system, the training data consists of portrait photos, each showing one person. In this scenario, unlearning is naturally performed on the level of data points. However, privacy issues can also arise at a different granularity of the data. For example, the leaked address of a celebrity may be widely circulated on social media, affecting features of hundreds of data points. The same problem occurs when training on emails and sensitive data, such as personal names or telephone numbers, appears in several emails during a conversation. Similarly, toxic content may be captured by language models from posts of various users over time.

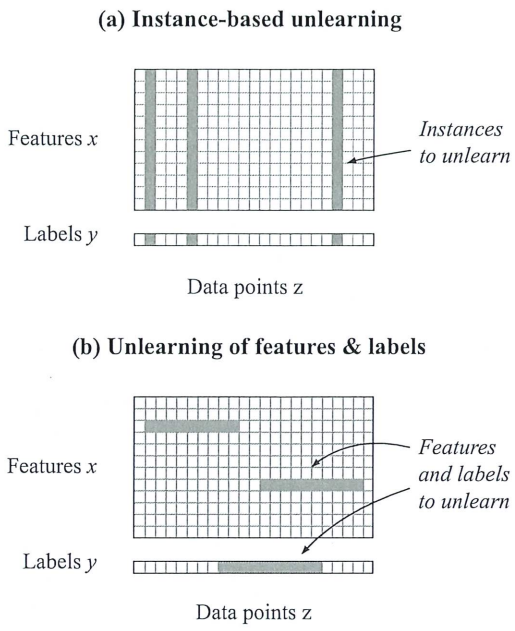

Fig. 2: Instance-based unlearning vs. unlearning of features and labels. The data to be removed is marked with orange.

In these cases, instance-based unlearning [3–8] is inefficient: First, the runtime advantage over retraining vanishes with the number of data points affected, as shown in the previous section. Second, removing entire instances and not just the affected features or labels unnecessarily degrades the performance of the corrected models. As a solution to this situation, we propose a method for unlearning of individual features and labels. As illustrated in Figure 2, this strategy operates on an orthogonal dimension of the data. Instead of correcting privacy issues along data instances (columns), we focus on resolving the issues in feature values and labels (rows). This is possible because we formulate unlearning as a closed-form update of the model, which enables us to correct features and labels at arbitrary positions in the training data.

To the best of our knowledge, we are the first to tackle the problem of unlearning from this perspective, thereby adding a new tool to the existing machinery for mitigating privacy threats in machine learning. In particular, our approach provides the following advantages:

- *Efficiency.* When privacy issues affect multiple data instances but are limited to particular features or labels, it is more efficient to correct these directly. As demonstrated in our evaluation, we achieve a significant performance improvement over existing approaches.
- *Flexibility.* Due to the concept of influence functions, we can correct arbitrary feature values and labels in the original training data. As a result, our approach can also unlearn entire data points, making it a versatile alternative to existing methods.

Revoking features. Based on appropriate definitions of Z and $\tilde{Z}$, our approach enables to replace the content of features and thus eliminate privacy leaks by overwriting sensitive data. In some scenarios, however, it might be necessary to even completely remove features from a learning model—a task that we denote as *revocation*. In contrast to the correction of features, this form of unlearning poses a unique challenge: The revocation of features reduces the input dimension of the model. While this adjustment can be easily carried out through retraining with adapted data, constructing a model update as in Equation (5) becomes tricky.

To address this problem, let us consider a model θ^* trained on a dataset $D \subset \mathbb{R}^d$. If we remove some features F from this dataset and train the model again, we obtain a new optimal model θ^*_{-F} with reduced input dimension. By contrast, if we set the values of the features F to zero in the dataset and train again, we obtain an optimal model $\theta^*_{F=0}$ with the same input dimension as θ^*. These two models are equivalent for a large class of learning models, including several neural networks as the following lemma shows.

Lemma 1. *For learning models processing inputs x using linear transformations of the form $\theta^T x$, we have $\theta^*_{-F} \equiv \theta^*_{F=0}$.*

Proof: It is easy to see that it is irrelevant for the dot product $\theta^T x$ whether a dimension of x is missing or equals zero in the linear transformation

$$\sum_{k:k\notin F} \theta[k]x[k] = \sum_{k} \theta[k]\mathbf{1}\{k \notin F\}x[k].$$

As a result, the loss $\ell(z,\theta) = \ell(\theta^T x, y, \theta)$ of both models is identical for every data point z. Hence, $L(\theta; D)$ is also equal for both models and thus the same objective is minimized during learning, resulting in equal model parameters. □

Lemma 1 enables us to erase features from many learning models by first setting them to zero, calculating the parameter update, and then reducing the input dimension of the models accordingly. Concretely, to revoke the features F from a learning model, we first locate all data points where these features are non-zero with

$$Z = \big\{(x,y) \in D : x[f] \neq 0, f \in F\big\}.$$

Then, we construct corresponding perturbations so that the features are set to zero with our approach,

$$\tilde{Z} = \big\{(x[f] = 0, y) : (x,y) \in Z,\ f \in F\big\}.$$

Finally, we adapt the input dimension by removing the affected inputs of the learning models, such as the corresponding neurons in the input layer of a neural network.

V. Update Steps for Unlearning

Our approach rests on changing the influence of training data in a closed-form update. In the following, we derive two strategies for calculating this closed form: a *first-order update* and a *second-order update*. The first strategy builds on the gradient of the loss function and thus can be applied to any model with differentiable loss. The second strategy incorporates second-order derivatives which limits the application to loss functions with an invertible Hessian matrix.

A. First-Order Update

Recall that we aim to find an update $\Delta(Z, \tilde{Z})$ that we can add to our model θ^* for unlearning. If the loss ℓ is differentiable, we can compute an optimal *first-order update* as follows

$$\Delta(Z,\tilde{Z}) = -\tau\Big(\sum_{\tilde{z}\in\tilde{Z}} \nabla_\theta \ell(\tilde{z},\theta^*) - \sum_{z\in Z} \nabla_\theta \ell(z,\theta^*)\Big) \quad (6)$$

where τ is a small constant that we refer to as *unlearning rate*. A complete derivation of Equation (6) is given in Appendix B1. Intuitively, this update shifts the model parameters from $\sum_{z\in Z} \nabla\ell(z,\theta^*)$ to $\sum_{\tilde{z}\in\tilde{Z}} \nabla\ell(\tilde{z},\theta^*)$ where the size of the update step is determined by the rate τ. This update strategy is similar to a gradient descent update GD given by

$$\mathrm{GD}(\tilde{Z}) = -\tau \sum_{\tilde{z}\in\tilde{Z}} \nabla_\theta \ell(\tilde{z},\theta^*).$$

However, it differs from this update step in that it moves the model to the *difference* in gradient between the original and perturbed data, which minimizes the loss on $\tilde{z}$ and at the same time removes the information contained in z.

The first-order update is a simple and yet effective strategy: The gradients of ℓ can be computed in $\mathcal{O}(p)$ [39] where p is the number of parameters in the learning model. However, the first-order update involves a parameter τ that controls the impact of the unlearning. To ensure that the data has been completely replaced, it is necessary to calibrate this parameter using a measure for the success of unlearning. In Section VII, we show how the exposure metric proposed by Carlini et al. [1] can be used for this calibration.

B. Second-Order Update

The unlearning rate τ can be eliminated if we make further assumptions on the properties of the loss ℓ. If we assume that ℓ is twice differentiable and strictly convex, the influence of a single data point can be approximated in closed form by

$$\frac{\partial \theta^*_{\epsilon,z\to\tilde{z}}}{\partial\epsilon}\bigg|_{\epsilon=0} = -H^{-1}_{\theta^*}\big(\nabla_\theta\ell(\tilde{z},\theta^*) - \nabla_\theta\ell(z,\theta^*)\big),$$

where $H^{-1}_{\theta^*}$ is the inverse Hessian of the loss at θ^*, that is, the inverse matrix of the second-order partial derivatives [see 21]. We can then perform a linear approximation as follows

$$\theta^*_{z\to\tilde{z}} \approx \theta^* - H^{-1}_{\theta^*}\big(\nabla_\theta\ell(\tilde{z},\theta^*) - \nabla_\theta\ell(z,\theta^*)\big). \quad (7)$$

Since all operations are linear, we can extend Equation (7) to multiple data points and finally obtain the *second-order update* for our approach:

$$\Delta(Z,\tilde{Z}) = -H^{-1}_{\theta^*}\Big(\sum_{\tilde{z}\in\tilde{Z}} \nabla_\theta \ell(\tilde{z},\theta^*) - \sum_{z\in Z} \nabla_\theta \ell(z,\theta^*)\Big). \quad (8)$$

A full derivation of this update step is provided in Appendix B2. Note that the update does not require a parameter calibration, since the parameter weighting of the changes is directly derived from the inverse Hessian of the loss function.

The second-order update is the preferred strategy for unlearning on models with a strongly convex and twice differentiable loss function that guarantee the existence of $H^{-1}_{\theta^*}$. Technically, the update step in Equation (8) can be easily calculated with common machine learning frameworks.

In order to obtain a small gradient residual norm for the first-order update the unlearning rate should be small, ideally in the order of $1/n\gamma$. Since $\|x_i\|_2 \leq 1$ we also have $m_j \ll 1$ if d is large and thus M acts as an additional damping factor for both updates when changing or revoking features.

Theorem 1 enables us to quantify the difference between unlearning and retraining. Concretely, if $\mathcal{A}(D')$ is an exact minimizer of L_b on D' with density $f_{\mathcal{A}}$ and $\mathcal{U}(\mathcal{A}(D), D, D')$ an approximated minimum obtained through unlearning with density $f_{\mathcal{U}}$, then Guo et al. [5] show that the max-divergence between $f_{\mathcal{A}}$ and $f_{\mathcal{U}}$ for the model θ produced by $\mathcal{U}$ can be bounded using the following theorem.

Theorem 2 (Guo et al. [5])**.** *Let $\mathcal{U}$ be an unlearning method with a gradient residual r with $\|r\|_2 \leq \epsilon'$. If the vector b is drawn from a probability distribution with density p satisfying that for any $b_1, b_2 \in \mathbb{R}^d$ there exists an $\epsilon > 0$ such that $\|b_1 - b_2\| \leq \epsilon'$ implies $e^{-\epsilon} \leq \frac{p(b_1)}{p(b_2)} \leq e^{\epsilon}$ then*

$$e^{-\epsilon} \leq \frac{f_{\mathcal{U}}(\theta)}{f_{\mathcal{A}}(\theta)} \leq e^{\epsilon}$$

for any θ produced by the unlearning method $\mathcal{U}$.

Theorem 2 equips us with a way to prove the certified unlearning property from Definition 1. Using the gradient residual bounds derived in Theorem 1, we can adjust the density function underlying the vector b so that Theorem 2 holds for both update steps of our unlearning approach.

Theorem 3. *Let $\mathcal{A}$ be the learning algorithm that returns the unique minimum of $L_b(\theta; D')$ and let $\mathcal{U}$ be an unlearning method that produces a model $\theta_{\mathcal{U}}$. If $\|\nabla L(\theta_{\mathcal{U}}; D')\|_2 \leq \epsilon'$ for some $\epsilon' > 0$ we have the following guarantees.*

1) *If b is drawn from a distribution with density $p(b) = e^{-\frac{\epsilon}{\epsilon'}\|b\|_2}$ then $\mathcal{U}$ performs ϵ-certified unlearning for $\mathcal{A}$.*
2) *If $p \sim \mathcal{N}(0, c\epsilon'/\epsilon)^d$ for some $c > 0$ then $\mathcal{U}$ performs (ϵ, δ)-certified unlearning for $\mathcal{A}$ with $\delta = 1.5e^{-c^2/2}$.*

Theorem 3 finally allows us to establish certified unlearning of features and labels in practice: Given a learning model with a bounded gradient residual norm and a privacy budget (ϵ, δ) we can calibrate the probability distribution of b to obtain a certified unlearning method.

Further details. We refer the reader for further details on certified unlearning to the appendix. In particular, we present the proofs for all theorems in Appendix D, we discuss the relation between our approach and differential privacy in Appendix E, and we show how the privacy budget (ϵ, δ) can support multiple unlearning requests in Appendix F.

VII. Empirical Analysis

We proceed with an empirical analysis of our approach and its capabilities. For this analysis, we examine the performance of unlearning in different scenarios and compare our method to other strategies for removing data, such as retraining, sharding, fine-tuning, and differentially private learning. As part of these experiments, we employ models with convex and non-convex loss functions to understand how this property affects the success of unlearning.

Unlearning scenarios. Our empirical analysis is based on three application scenarios in which sensitive and personal information need to be removed from learning models.

Scenario 1: Sensitive features. Our first scenario deals with linear models for classification. These models are widely used in fraud, spam and malware detection due to their simplicity and strongly convex loss function [40, 41]. While they induce a slight performance drop compared to neural networks (see Appendix H), they still remain of practical relevance. We investigate how our approach can unlearn sensitive features from these models (see Section VII-A).

Scenario 2: Unintended memorization. In the second scenario, we consider the problem of unintended memorization [1]. Language models based on recurrent neural networks can accidentally memorize sensitive data, such as credit card numbers or private messages. Through specifically crafted inputs, an attacker can extract this data during text completion [1, 15]. We apply unlearning of features and labels to remove these privacy leaks from language models (see Section VII-B).

Scenario 3: Data poisoning. For the third scenario, we focus on poisoning attacks in computer vision. Here, an adversary aims at misleading an object recognition task by flipping a few labels of the training data. The label flips significantly reduce the performance of the learning model. We use unlearning of labels as a strategy to correct this defect and restore the original performance without retraining (see Section VII-C).

Performance measures. The success of unlearning depends on three properties: An effective method must (1) remove the selected data, (2) preserve the model's quality, and (3) be efficient compared to retraining. A method that fails to satisfy any of these properties is ineffective, because it either does not correctly unlearn data, degrades the model, or lacks behind retraining. To reflect this setting, we introduce three performance measures for our empirical analysis.

Efficacy of unlearning. The most important property for successful unlearning is the removal of data. While certified unlearning ensures this removal, we cannot provide similar guarantees for models with non-convex loss functions. As a result, we need to employ measures that quantitatively assess the *efficacy* of unlearning. For example, we can use the *exposure metric* [1] to measure the memorization of specific sequences in language models after unlearning.

Fidelity of unlearning. The second property contributing to the success of unlearning is the performance of the corrected model, which we denote as *fidelity*. An unlearning method is of practical use only if it keeps the performance as close as possible to the original model. Hence, we consider the fidelity as the second performance measure. In our experiments, we use the loss and accuracy of the original and corrected model on a hold-out set to determine this property.

Efficiency of unlearning. If the training data used to generate a model is available, a simple unlearning strategy is retraining. This strategy, however, involves significant runtime and storage costs. Therefore, we consider the *efficiency* of unlearning as the third property. In our experiments, we measure the runtime and the number of gradient calculations for each unlearning method on the datasets of the three scenarios.

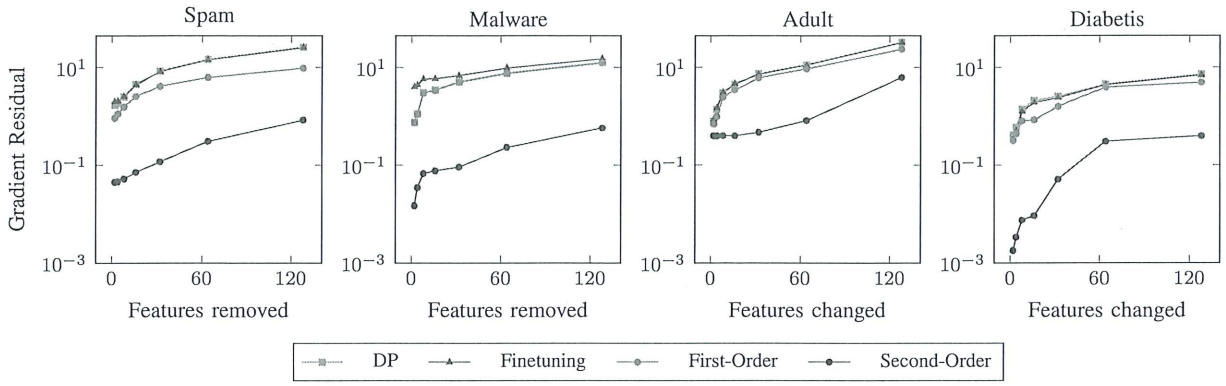

Fig. 4: Efficacy (gradient residual) of the certified unlearning methods for varying number of affected features (Lower values are better).

However, the steepness of this development gradually reduces as more features are remove or changed. Our second-order update significantly outperforms all other methods in this scenario. The gradient residual norms are an order of magnitude lower on the Spam, Malware, and Diabetis dataset, regardless of the amount of affected features. Among the other unlearning methods, no clear ranking can be determined in this experiment.

Fidelity evaluation. We evaluate the fidelity of the unlearning methods using two techniques: First, we investigate the loss between retraining and unlearning on the test data as proposed by Koh and Liang [10]. Figure 5 shows this comparison for the Diabetis and Malware dataset when removing or replacing 100 features, respectively. We observe that the second-order update approximates the retraining very well, since the points are close to the diagonal line. In contrast, the other methods cannot always adapt to the distribution shift, resulting in larger differences. This trend also holds for the other datasets which we report in Appendix G1.

Second, we use the test accuracy of a model that provides certified unlearning for the removal or replacement of features to evaluate the fidelity. To simulate a realistic application of certified unlearning, we fix a privacy budget (ϵ, δ) in advance and adapt the noise term b in relation to the amount of affected features. That is, the more features are changed, the more we need to increase the noise on the model to achieve the same guarantees. In particular, Theorem 3 states that the noise term b must be sampled from a Gaussian normal distribution with variance σ, which is given by

$$\sigma = \frac{\beta c}{\epsilon}, \qquad \text{where } c = \sqrt{2\log(1.5/\delta)}. \tag{9}$$

where β is constant corresponding to a general upper bound of the gradient residual loss for the considered learning task.

In the following, we select $\epsilon = 0.1$ and $\delta = 0.01$ as a privacy budget, which yields $c \approx 3.16$. We compute the bound β on the gradient residual by sampling 100 feature combinations to unlearn, compute σ and determine the resulting test accuracy of the four datasets. As we see in the following, this privacy budget is strict and limits the amount of features that can be adapted. Nevertheless, if a larger number of features needs to be removed, this budget can be increased, though at the cost of weakening the certification guarantees.

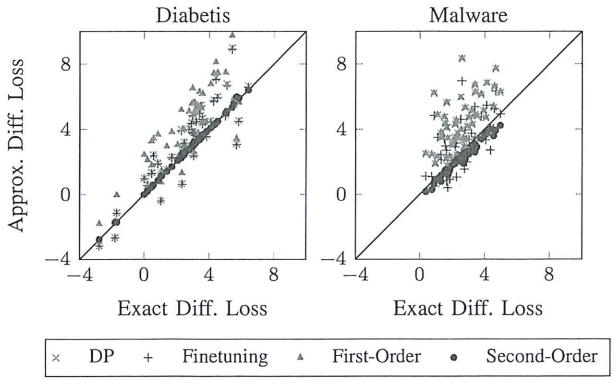

Fig. 5: Difference in loss between retraining and unlearning with 100 affected features.

The accuracy of the models is shown in Figure 6 for a varying number of removed or replaced features, respectively. The accuracy reduces with the amount of affected features, as the noise on the model weights is increased accordingly. While for the low-dimensional datasets this reduction is moderate, we observe a strong decline of fidelity for the high-dimensional data. This decline results from the privacy budget that requires a notable amount of noise to be added to enable a certified removal of entire dimensions.

Our second-order update shows the best performance of all methods and remains close to retraining if up to 60 features are changed. In contrast, the other methods quickly drop in accuracy already when unlearning 20 or less features. An exception is sharding. While the method provides the weakest performance on the high-dimensional datasets due to instability in the majority voting, it is almost identical to retraining on the low-dimensional datasets.

Efficiency evaluation. Finally, we evaluate the efficiency of the different methods. Table II shows the measured runtime on the Malware dataset, while the results for the other datasets are shown in Appendix G2. We omit measurements for the differential privacy baseline, as it does not involve an unlearning step. Due to the simple structure of the logistic regression, the

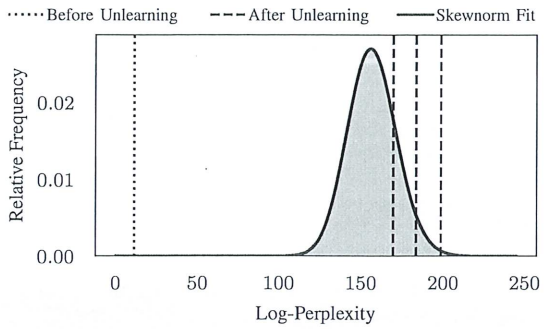

Fig. 7: Perplexity distribution of the language model. The vertical lines indicate the perplexity of an inserted telephone number. Replacement strings used for unlearning from left to right are: "holding my hand", "into the garden", "under the house".

of the total 10^{15} possible sequences in Q. The perplexity of the inserted number differs significantly from all other number combinations in Q (dashed line to the left), indicating that it has been memorized by the underlying language model. After unlearning with different replacements, the number moves close to the center of the distribution (dashed lines to the right).

Unlearning task. To unlearn the memorized sequences, we replace each digit of the telephone number in the data with a different character, such as a random or constant value. Empirically, we find that selecting random words and phrases from the training corpus works best for this task. Some examples of replacements are shown in Table IV. The model has already captured these character dependencies, resulting in small updates of the model parameters. The unlearning of the substitutions, however, is more involved than in the previous scenario. The language model is trained to predict a character from preceding characters. Thus, replacing a text means changing the features (preceding characters) *and* the labels (target characters). Therefore, we combine both changes in a single set of perturbations in this setting.

Efficacy evaluation. First, we evaluate whether the memorized numbers have been successfully unlearned from the language model. An important result of the study by Carlini et al. [1] is that the exposure is associated with an extraction attack: For a set Q with r elements, a sequence with an exposure smaller than r cannot be extracted. Consequently, we test three different substitution sequences for each telephone number, calculate the exposure metric, and use the best for our evaluation. Table III shows the results of this experiment.

TABLE III: Exposure metric of the canary sequence for different lengths. Lower exposure values make extraction harder.

Number length	5	10	15	20
Original model	43 ± 19	70 ± 26	109 ± 16	99 ± 52
Retraining	0 ± 0	0 ± 0	0 ± 0	0 ± 0
Fine-tuning	39 ± 21	31 ± 44	50 ± 50	57 ± 73
SISA (n shards)	0 ± 0	0 ± 0	0 ± 0	0 ± 0
First-Order	0 ± 0	0 ± 0	0 ± 0	0 ± 0
Second-Order	0 ± 0	0 ± 0	0 ± 0	0 ± 0

We observe that our first-order and second-order updates yield exposure values close to zero (< 0.001) for all sequence lengths, rendering an extraction impossible. Retraining and SISA yield an exposure of zero by design since the injected sequences are removed from the training data. In contrast, fine-tuning leaves a large exposure in the model, so that a successful extraction is still possible. On closer inspection, we find that the performance of fine-tuning depends on the order of the training data, resulting in high deviation in the experimental runs. This problem cannot be easily mitigated by learning over further epochs and thus highlights the need for unlearning techniques.

We also find that the selected substitution plays an important role for unlearning. In Figure 7, we report the log-perplexity of the canary for three different substitutions after unlearning. Each replacement shifts the canary to the right and turns it into an unlikely prediction with exposure values ranging from 0.01 to 0.3. While we use the replacement with the lowest exposure in our experiments, the other substitution sequences would also impede a successful extraction.

It remains to show what the model actually predicts after unlearning when given the canary sequence. Table IV shows different completions of the canary sentence after unlearning with our second-order update and replacement strings of different lengths. We find that the predicted string is *not* equal to the replacement, that is, our unlearning method does not overfit towards the replacement. The sentences follow the language structure and reflect the wording of the novel. Both observations indicate that the parameters of the language model are indeed corrected and not just overwritten with other values.

Fidelity evaluation. To evaluate the fidelity, we examine the accuracy of the corrected models as shown in Figure 8 (left), where the accuracy of all unlearning methods is plotted for different numbers of affected data points. For small changes, all approaches except sharding come close to retraining in performance. Sharding is unsuited for unlearning in this scenario. The method uses an ensemble of sub-models trained on different shards. Each sub-model produces an own sequence of text and thus combining them with majority voting leads to inaccurate predictions.

With larger changes to the model, the accuracy of both of our methods gradually begins to decline. The second-order update provides slightly better results because the Hessian contains

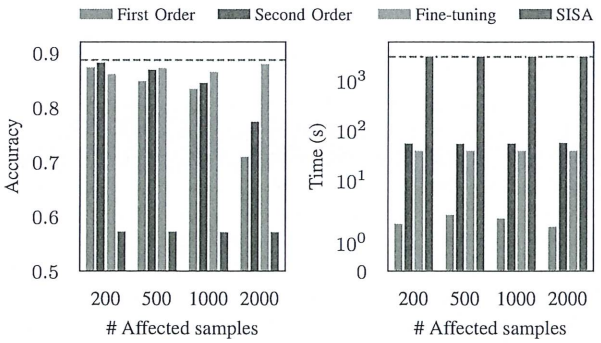

Fig. 8: Accuracy after unlearning unintended memorization. The dashed line corresponds to a model retrained from scratch.

order update as well as fine-tuning, which all come close to the original performance for 2,500 poisoned labels. However, we observe a continuous performance decline when more labels are poisoned. A manipulation of 10,000 labels during training cannot be sufficiently reverted by any of the methods. Interestingly, sharding is the only exception here. Although the method provides the lowest performance in this experiment, it is not affected by the number of poisoned labels, as all shards are simply retrained with the corrected labels.

Efficiency evaluation. Lastly, we evaluate the runtime of each approach to quantify its efficiency. The experiments are executed on the same hardware as the previous ones. In contrast to the language model, however, we are able to perform all calculations on the GPU which allows for a fair comparison. Figure 9 shows that the first-order update and fine-tuning are very efficient and can be computed in approximately 10 seconds, whereas retraining requires over 15 minutes. The second-order update is slightly slower but still with 20.8 seconds two orders of magnitude faster than retraining. In contrast, the sharding approach is the slowest and does not provide any advantage over retraining. Consequently, the first-order update and fine-tuning provide the best strategies for unlearning in this scenario.

In computer vision, learning models are often huge. Hence, we also investigate the scalability of our approach. Figure 10 shows the accuracy and runtime of the first-order and second-order update for increasing model sizes. In particular, we scale the number of model parameters from 1.8 millions up to 42 millions. For both update techniques, the accuracy remains roughly the same. The runtime naturally increases with the model size, yet the slope is (almost) linear for both approaches. We observe a slight peak when reaching the limits of our hardware. Overall, we find that the linear runtime bounds of the underlying algorithm hold in practice [47].

> *Takeaway message.* Label poisoning can be mitigated by unlearning labels. Our first-order update and fine-tuning are suitable methods and provide the best trade-off between efficacy (=fidelity) and efficiency.

VIII. Limitations

Although our approach successfully removes features and labels in different experiments, it obviously has limitations that need to be considered in practice.

→ GOOD FOR THOUSANDS, LESS GOOD FOR MILLIONS

Limits of unlearning. The efficacy of unlearning decreases with the number of affected features and labels. While privacy leaks with hundreds of sensitive features and thousands of labels can be handled well with our approach, changing millions of data points exceeds its capabilities. If our approach allowed to correct changes of arbitrary size, it could be used as a "drop-in" replacement for all learning algorithms—which obviously is impossible [48]. Nevertheless, our method offers a significant speed-up over retraining and sharding in situations where a moderate number of data points needs to be corrected.

Non-convex loss functions. Our approach can only guarantee certified unlearning for strongly convex loss functions that have Lipschitz-continuous gradients. While both update steps of our approach work well for neural networks with non-convex functions, as we demonstrate in the empirical evaluation, they require an additional measure to validate unlearning success. Fortunately, such external measures are often available, as they typically provide the basis for characterizing data leakage prior to its removal. Similarly, we use the fidelity to measure how our approach corrects a poisoning attack.

→ GUARANTEES OF STRONGLY CONVEX LOSS FUNCTION.
EMPIRICALLY PROVEN TO WORK FOR NON CONVEX LOSS FUNCTION TOO (LLM SPECIFICALLY)

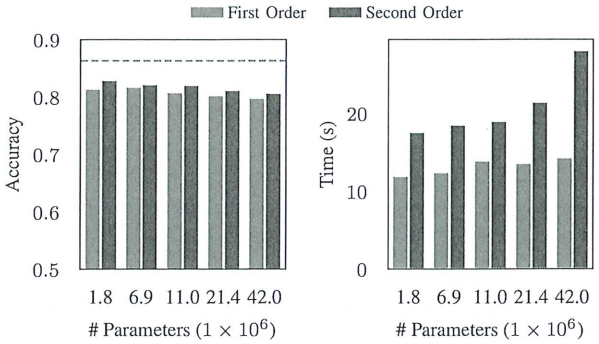

Fig. 10: Accuracy (left) and runtime (right) on held-out data after unlearning 5,000 poisoned labels for multiple model sizes. The dashed line corresponds to the accuracy of the models on clean data.

→ ONLY FIX, BUT DOESN'T RECOGNIZE DATA LEAKS

Unlearning requires detection. Finally, we point out that our method requires knowledge of the data to be removed. Detecting privacy leaks in learning models is a hard problem outside of the scope of this work. The nature of privacy leaks depends on the considered data, learning models, and application. For example, the analysis of Carlini et al. [1, 15] focuses on sequential data in generative learning models and cannot be easily transferred to other learning models or image data. As a result, we limit this work to repairing leaks rather than finding them.

IX. Conclusion

Instance-based unlearning is concerned with removing data points from a learning model *after* training—a task that becomes essential when users demand the "right to be forgotten". However, sensitive information is often spread across instances, impacting larger portions of the training data. Instance-based unlearning is limited in this setting. As a remedy, we propose a novel framework for unlearning features and labels based on the concept of influence functions. Our approach captures the changes to a learning model in a closed-form update, providing significant speed-ups over other approaches.

We demonstrate the effectivity of our approach in a theoretical and empirical analysis. Based on the concept of differential privacy, we prove that our framework enables certified unlearning on models with a strongly convex loss function and evaluate the benefits of our unlearning strategy in three practical scenarios. In particular, for generative language models, we are able to remove unintended memorization while preserving the functionality of the models.

We hope that this work fosters further research on machine unlearning and sharpens theoretical bounds on privacy in machine learning. To support this development, we make all our implementations and datasets publicly available at https://github.com/alewarne/MachineUnlearning.

S. Soatto, "Mixed-privacy forgetting in deep networks," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.
[32] A. Golatkar, A. Achille, and S. Soatto, "Eternal sunshine of the spotless net: Selective forgetting in deep networks," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
[33] K. R. Rad and A. Maleki, "A scalable estimate of the extra-sample prediction error via approximate leave-one-out," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 82, 2018.
[34] J. Wakefield, "Microsoft chatbot is taught to swear on Twitter," BBC News, https://www.bbc.com/news/technology-35890188.
[35] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*, 2nd ed. John Wiley & Sons, 2000.
[36] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Foundations and Trends in Theoretical Computer Science*, p. 211–407, 2014.
[37] A. Graves, "Generating sequences with recurrent neural networks," Computing Research Repository (CoRR), Tech. Rep. arXiv:1308.0850, 2013.
[38] I. Sutskever, J. Martens, and G. Hinton, "Generating text with recurrent neural networks," in *International Conference on Machine Learning (ICML)*, 2011.
[39] B. A. Pearlmutter, "Fast exact multiplication by the hessian," *Neural Computation*, vol. 6, no. 1, p. 147–160, 1994.
[40] J. Attenberg, K. Weinberger, A. Dasgupta, A. Smola, and M. Zinkevich, "Collaborative email-spam filtering with the hashing trick," in *Conference on Email and Anti-Spam (CEAS)*, 2009.
[41] D. Arp, M. Spreitzenbarth, M. Hübner, H. Gascon, and K. Rieck, "Drebin: Efficient and explainable detection of Android malware in your pocket," University of Göttingen, Tech. Rep. IFI-TB-2013-02, Aug. 2013.
[42] V. Metsis, G. Androutsopoulos, and G. Paliouras, "Spam filtering with naive bayes - which naive bayes?" in *Proc. of Conference on Email and Anti-Spam (CEAS)*, 2006.
[43] D. Dua and C. Graff, "UCI machine learning repository. diabetis data set." 2017. [Online]. Available: http://archive.ics.uci.edu/ml
[44] ——, "UCI machine learning repository. census income data set." 2017. [Online]. Available: http://archive.ics.uci.edu/ml
[45] S. Merity, N. S. Keskar, and R. Socher, "An analysis of neural language modeling at multiple scales," *arxiv:1803.08240*, 2018.
[46] H. Xiao, H. Xiao, and C. Eckert, "Adversarial label flips attack on support vector machines," in *Proc. of European Conference on Artificial Intelligence (ECAI)*, 2012, pp. 870–875.
[47] N. Agarwal, B. Bullins, and E. Hazan, "Second-order stochastic optimization for machine learning in linear time," *Journal of Machine Learning Research (JMLR)*, p. 4148–4187, 2017.
[48] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 67, 1997.
[49] S. Boyd and L. Vandenberghe, *Convex Optimization*, 2004.
[50] D. Desfontaines and B. Pejó, "Sok: Differential privacies," *Proceedings on Privacy Enhancing Technologies*, pp. 288–313, 2020.
[51] D. Kifer and A. Machanavajjhala, "No free lunch in data privacy," in *ACM International Conference on Management of Data*, 2011, p. 193–204.
[52] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *ACM Conference on Computer and Communications Security (CCS)*, 2016, pp. 308–318.
[53] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Conference on Theory of Cryptography*, 2006, p. 265–284.
[54] C. Dwork, G. N. Rothblum, and S. Vadhan, "Boosting and differential privacy," in *Annual Symposium on Foundations of Computer Science*, 2010, p. 51–60.
[55] C. Dwork and J. Lei, "Differential privacy and robust statistics," in *Annual ACM Symposium on Theory of Computing*, 2009, p. 371–380.
[56] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009.
[57] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.

APPENDIX

A. Stochastic Analysis of Sharding

To better understand the need for closed-form updates on model parameters, we examine current sharding strategies and investigate the circumstances under which they reach their limits. Bourtoule et al. [4] propose an unlearning method with the core idea to train separate models on distinct parts of training data. While the authors discuss limitations of their approach like performance degradation, we perform a stochastic analysis to find upper bounds for the number of affected data points at which sharding becomes as efficient as retraining.

In this context, we consider n data instances to unlearn which are uniformly distributed across s shards. Let $p(n)$ denote the probability that all shards contain at least one of these samples which leads to the worst-case scenario of having to retrain all shards. Since calculating $p(n)$ as stated above is difficult, we reformulate the task to solve an equivalent problem: We seek the probability $\hat{p}_k(n)$ that at most k shards remain unaffected by any sample. We set $k = s$ such that $\hat{p}_s(n)$ indicates the probability that any combination of $i \in \{1, \dots, s\}$ shards are unaffected. If this probability is zero there are no unaffected shards. Hence, this corresponds to the inverse of our target probability $p(n) = 1 - \hat{p}_s(n)$.

To calculate $\hat{p}_s(n)$, we first determine the probability of exactly i shards to remain unaffected. In general, there are $(s-i)^n$ combinations to distribute n samples on the dataset excluding i shards. Since there are $\binom{s}{i}$ possible ways to select the i shards to be left out, the total number of combinations is given by $\binom{s}{i}(s-i)^n$. However, we cannot simply sum these terms up for different values of i since the unaffected shards in the combinations partly overlap. To account for this, we apply the inclusion-exclusion principle and finally divide the adjusted term by the number of combinations including all shards:

$$\hat{p}_s(n) = \frac{\sum_{i=1}^{s}(-1)^{i+1}\binom{s}{j}(s-i)^n}{s^n}$$

Algorithm 1: Parameter update

Input: model θ^*, loss functions L and ℓ, order o, unlearning rate τ, batch-size B, iterations m, damping d, scale s, repetitions r
Output: Parameter update $\Delta(Z, \tilde{Z})$
Data: D, D', Z, $\tilde{Z}$

$g_1 = \sum_{\tilde{z}\in\tilde{Z}} \nabla_\theta \ell(\tilde{z}, \theta^*)$, $g_2 = \sum_{z\in Z} \nabla_\theta \ell(z, \theta^*)$
$v = g_1 - g_2$
if $o == 1$ **then**
  $\Delta = -\tau v$
else
  $\Delta = 0$
  for *i=1:r* **do**
    $\theta_{\text{new}} = 0$
    for *j=1:m* **do**
      batch = sample(D', size=B)
      hvp $= \nabla_\theta\big(v^T \nabla_\theta L(\text{batch}, \theta^*)\big)$
      $\theta_{\text{new}} = v + (1-d)\theta_{\text{new}} - \text{hvp}/s$
    $\Delta = \Delta + \theta_{\text{new}}/r$
return Δ

Averaging batches of data points further speeds up the approximation. Choosing t large enough so that the updates converge and averaging r runs to reduce the variance of the results, we obtain $\tilde{H}_t^{-1} v$ as our final estimate of $H^{-1}v$ in $\mathcal{O}(rtp)$ of time. The pseudo-code in Algorithm 1 summarizes how we compute the second-order update.

D. Proofs for Certified Unlearning

We continue to present the proofs for certified unlearning of our approach and, in particular, the bounds of the gradient residual used in Section VI. First, let us recall Theorem 1 from Section VI-A.

Theorem 1. *Assume that $\|x_i\|_2 \leq 1$ for all data points and the gradient $\nabla\ell(z, \theta)$ is γ_z-Lipschitz with respect to z at θ^* and γ-Lipschitz with respect to θ. Further let $\tilde{Z}$ change the features $j, \ldots, j+F$ by magnitudes at most $m_j, \ldots, m_{j+F}$. If $M = \sum_{j=1}^{F} m_j$ the following upper bounds hold:*

1) For the first-order update of our approach, we have

$$\big\|\nabla L\big(\theta^*_{Z\to\tilde{Z}}, D'\big)\big\|_2 \leq (1 + \tau\gamma n)\gamma_z M|Z|$$

2) If $\nabla^2\ell(z, \theta)$ is γ''-Lipschitz with respect to θ, we have

$$\big\|\nabla L\big(\theta^*_{Z\to\tilde{Z}}, D'\big)\big\|_2 \leq \gamma''\Big(\frac{M\gamma_z}{\lambda}\Big)^2 n|Z|^2$$

for the second-order update of our approach.

To prove this theorem, we begin by introducing a small lemma which is useful for investigating the gradient residual of the optimal learning model θ^* on a dataset D'.

Lemma 2. *Given a radius $R > 0$ with $\|\delta_i\|_2 \leq R$, a gradient $\nabla\ell(z, \theta)$ that is γ_z-Lipschitz with respect to z, and a learning model θ^*, we have*

$$\big\|\nabla L\big(\theta^*, D'\big)\big\|_2 \leq R\gamma_z|Z|.$$

Proof: By definition, we have

$$\nabla L(\theta^*; D') = \sum_{z\in D'} \nabla\ell\big(z, \theta^*\big) + \lambda\theta^*.$$

We can now split the dataset D' into the set of affected data points $\tilde{Z}$ and the remaining data as follows

$$\begin{aligned}\nabla L(\theta^*; D') &= \sum_{z\in D'\setminus\tilde{Z}} \nabla\ell\big(z, \theta^*\big) + \sum_{\tilde{z}\in\tilde{Z}} \nabla\ell\big(\tilde{z}, \theta^*\big) + \lambda\theta^* \\ &= \sum_{z\in D\setminus Z} \nabla\ell\big(z, \theta^*\big) + \sum_{\tilde{z}\in\tilde{Z}} \nabla\ell\big(\tilde{z}, \theta^*\big) + \lambda\theta^*.\end{aligned}$$

By applying a zero addition and leveraging the optimality of θ^* on D, we then express the gradient as follows

$$\nabla L(\theta^*; D') = 0 + \sum_{z_i\in Z} \nabla\ell\big(z_i + \delta_i, \theta^*\big) - \nabla\ell\big(z_i, \theta^*\big). \quad (13)$$

Finally, using the Lipschitz continuity of $\nabla\ell$, we get

$$\begin{aligned}\big\|\nabla L\big(\theta^*, D'\big)\big\|_2 &\leq \sum_{z_i\in Z} \big\|\nabla\ell\big(z_i + \delta_i, \theta^*\big) - \nabla\ell\big(z_i, \theta^*\big)\big\|_2 \\ &\leq \sum_{x_i, y_i\in Z} \gamma_z\big\|\delta_i\big\|_2 \leq M\gamma_z|Z|.\end{aligned}$$

□

We proceed to prove the update bounds of Theorem 1. The proof is structured in two parts, where we start with investigating the first case and then proceed with the second case of the theorem.

Proof: (Case 1) For the first-order update, we recall that

$$\theta^*_{Z\to\tilde{Z}} = \theta^* - \tau G(Z, \tilde{Z})$$

where $\tau \geq 0$ is the unlearning rate and we have

$$G(Z, \tilde{Z}) = \sum_{z_i\in Z} \nabla\ell(z_i + \delta_i, \theta) - \nabla\ell(z_i, \theta)$$

Consequently, we seek to bound the norm of

$$\nabla L\big(\theta^*_{Z\to\tilde{Z}}, D'\big) = \nabla L\big(\theta^* - \tau G(Z, \tilde{Z}), D'\big).$$

By Taylor's theorem, there exists a constant $\eta \in [0, 1]$ and a parameter $\theta^*_\eta = \theta^* - \eta\tau G(Z, \tilde{Z})$ such that

$$\begin{aligned}\nabla L\big(\theta^*_{Z\to\tilde{Z}}, D'\big) &= \nabla L\big(\theta^*, D'\big) \\ &\quad + \nabla^2 L\big(\theta^* + \eta(\theta^*_{Z\to\tilde{Z}} - \theta^*), D'\big)\big(\theta^*_{Z\to\tilde{Z}} - \theta^*\big) \\ &= \nabla L\big(\theta^*, D'\big) - \tau H_{\theta^*_\eta} G(Z, \tilde{Z}).\end{aligned}$$

In the proof of Lemma 2 we show that $\nabla L\big(\theta^*, D'\big) = G(Z, \tilde{Z})$ and thus we get

$$\begin{aligned}\big\|\nabla L\big(\theta^*_{Z\to\tilde{Z}}, D'\big)\big\|_2 &= \|G(Z, \tilde{Z}) - \tau H_{\theta^*_\eta} G(Z, \tilde{Z})\|_2 \\ &= \big\|\big(I - \tau H_{\theta^*_\eta}\big)G(Z, \tilde{Z})\big\|_2 \\ &\leq \big\|I - \tau H_{\theta^*_\eta}\big\|_2 \big\|G(Z, \tilde{Z})\big\|_2.\end{aligned}$$

Due to the γ-Lipschitz continuity of the gradient $\nabla\ell$, we have $\|H_{\theta^*_\eta}\|_2 \leq n\gamma$ and thus

$$\|I - \tau H_{\theta^*_\eta}\|_2 \leq 1 + \tau\gamma n$$

TABLE V: Average runtime when removing 100 random combinations of features for the different datasets.

	Spam			Diabetis			Adult		
Method	**Gradients**	**Runtime**	**Speed-up**	**Gradients**	**Runtime**	**Speed-up**	**Gradients**	**Runtime**	**Speed-up**
Retraining	1.4×10^6	1.52s	—	1.1×10^4	6.71ms	—	5.1×10^6	2.94s	—
SISA (5 shards)	4.1×10^5	0.56s	2.7×	9.8×10^3	29.55ms	0.2×	2.4×10^6	1.65s	1.8×
Fine-tuning	2.6×10^4	0.80s	1.9×	6.1×10^2	0.60ms	11.2×	3.9×10^4	0.09s	32.7×
First-Order	1.1×10^4	0.04ms	38.0×	1.0×10^2	0.10ms	67.1×	1.0×10^2	4.87ms	603.7×
Second-Order	6.5×10^4	5.42s	0.3×	1.3×10^4	0.23ms	29.2×	7.8×10^4	0.03s	98.0×

each intermediate step, which increases the computational effort. An empirical comparison of our approach with multiple update steps is presented in Appendix G3.

In terms of unlearning certifications, we can extend Theorem 1 and show that the gradient residual bound after T update steps remains smaller than TC, where C is the bound of a single step: If θ_t is the t-th solution obtained by our updates with gradient residual r_t then θ_t is an exact solution of the loss function $L_b(\theta, D') - r_t^T\theta$ by construction. This allows applying Theorem 1 to each θ_t and obtain the bound TC by the triangle inequality. Therefore, the gradient residual bound rises linearly in the number of applications of $\mathcal{U}$. This result is in line with the composition theorem of differential privacy research [53–55] which states that applying an ϵ-DP algorithm n times results in a $n\epsilon$-DP algorithm.

G. Evaluation of Certified Unlearning

1) Fidelity evaluation with loss: In Section VII, we evaluate the fidelity of the retrained model with different approaches using the difference in test loss presented in a scatter plot. Figure 11 shows the plots for the Adult and Spam dataset which we omitted previously due to space limitations. As for the other datasets, it is apparent that the second-order update has the highest correlation with the retrained model, i.e., the points are closest to the identity line. We also see less variance in terms of deviation from the identity line compared to the other approaches.

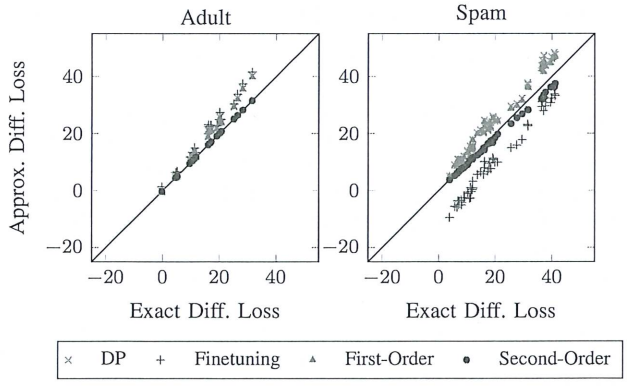

Fig. 11: Difference in test loss between retraining and unlearning when removing or changing random combinations of features. For perfect unlearning the results would lie on the identity line.

2) Efficiency evaluation: In Section VII, we present the runtime evaluation only for the Malware dataset. We show the evaluations for the remaining datasets in Table V. The entries are based on the experiments regarding fidelity where we removed or replaced 100 combinations of 100 features from the datasets. It can be seen that the computation of the inverse Hessian is faster than retraining in case of the Diabetis and Adult datasets making the second-order update very efficient in these cases.

3) Sequential unlearning steps: So far, we have presented our approach as one-shot unlearning, that is, all perturbed samples are collected in the set $\tilde{Z}$ and the update is performed on one step. The theoretical analysis in Section VI, however, shows that the error in approximation and the gradient residual norm rise with the size of $\tilde{Z}$. Therefore, a practitioner might want to perform the updates in smaller portions to keep the error in each update small. To evaluate this setting, we repeat the previous experiment but now split the update into 10 steps of equal size.

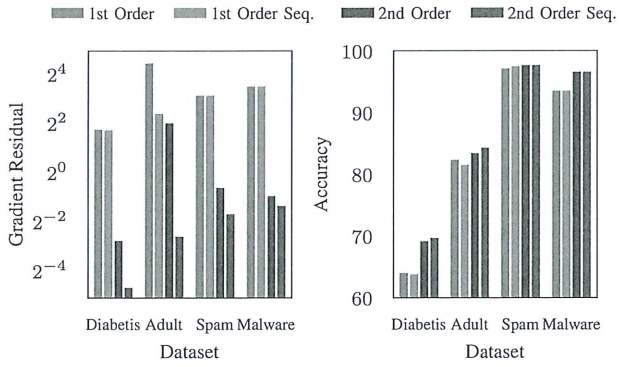

Fig. 12: Average accuracy of the corrected models (in %) when removing 100 features at once or sequentially in 10 steps.

Figure 12 shows the comparison between sequential and one-shot updates regarding the gradient residual norm and accuracy on test data. In terms of accuracy, the sequential updates give only slight performance increases for both methods. For the gradient residual, however, we observe strong decreases when applying the updates sequentially, especially for the Diabetis and Adult dataset. This confirms the results of Theorem 1 empirically and presents a special working mode of our approach. As pointed out in Appendix F, this increase in privacy budget comes with an increase in runtime for the second-order