MUter: Machine Unlearning on Adversarially Trained Models

Junxu Liu[1,*], Mingsheng Xue[2,*], Jian Lou[3,6,†], Xiaoyu Zhang[4], Li Xiong[5], Zhan Qin[3,6]
[1]Renmin University of China [2]Guangzhou Institute of Technology, Xidian University [3]Zhejiang University
[4]Xidian University [5]Emory University [6]ZJU-Hangzhou Global Scientific and Technological Innovation Center
junxu_liu@ruc.edu.cn xuemingsheng@stu.xidian.edu.cn jian.lou@zju.edu.cn
xiaoyuzhang@xidian.edu.cn lxiong@emory.edu qinzhan@zju.edu.cn

Abstract

Machine unlearning is an emerging task of removing the influence of selected training datapoints from a trained model upon data deletion requests, which echoes the widely enforced data regulations mandating the Right to be Forgotten. Many unlearning methods have been proposed recently, achieving significant efficiency gains over the naive baseline of retraining from scratch. However, existing methods focus exclusively on unlearning from standard training models and do not apply to adversarial training models (ATMs) despite their popularity as effective defenses against adversarial examples. During adversarial training, the training data are involved in not only an outer loop for minimizing the training loss, but also an inner loop for generating the adversarial perturbation. Such bi-level optimization greatly complicates the influence measure for the data to be deleted and renders the unlearning more challenging than standard model training with single-level optimization. This paper proposes a new approach called MUter for unlearning from ATMs. We derive a closed-form unlearning step underpinned by a total Hessian-related data influence measure, while existing methods can mis-capture the data influence associated with the indirect Hessian part. We further alleviate the computational cost by introducing a series of approximations and conversions to avoid the most computationally demanding parts of Hessian inversions. The efficiency and effectiveness of MUter have been validated through experiments on four datasets using both linear and neural network models.

1. Introduction

Machine learning (ML) models are increasingly applied to a broad range of applications, accompanied by growing concerns about their privacy and robustness issues. Both issues are actively studied in recent years [45]. On the privacy side, model inversion attack [16] and membership inference attack [51] reveal that the trained ML model contains sensitive information of its training data, which can cause privacy loss for individuals contributing their data for model training. On the robustness side, adversarial example attack is one of the most well-recognized robustness attacks [24, 30, 39, 62]. It can easily fool an undefended model, e.g., standard training model (STM), to misclassify by a small adversarial perturbation on the input [9, 24, 39]. Many works focus either on the privacy [1, 35, 37, 42, 51] or on the robustness aspect [9, 33, 38, 53–56, 66], few works studied both. Yet, it is critical to consider privacy and robustness jointly [20, 27, 36, 41, 47, 48, 52] to build ML models to simultaneously meet data privacy regulations and ensure robustness against adversarial threats.

In this paper, we target a new joint privacy-robustness problem to simultaneously meet emerging privacy regulations and ensure adversarial robustness of the model, which has not been examined before: *how to efficiently and effectively remove the influence of a training datapoint from an adversarially trained model upon data deletion request?*

The privacy need is driven by the widely enacted user data regulations that enforce the Right to be Forgotten, for example, the European Union's GDPR [17], the California Consumer Privacy Act (CCPA), and Canada's proposed Consumer Privacy Protection Act (CPPA). These regulations mandate the deletion of personal data upon user requests and can even include the deletion of models and algorithms derived from the user data, e.g., the Federal Trade Commission [15]. Machine unlearning [5, 8] aims to obtain an updated model with the influence of the target datapoint removed in an effective and efficient way. That is, the updated model should be similar to the model obtained by the computationally expensive retraining-from-scratch approach, while consuming less computation [21–23, 26, 31, 43, 44, 58, 60, 61, 65].

The robustness is achieved by the adversarial training model (ATM) [3, 24, 30, 39, 59], which is a popular and effective defense for enhancing the model robustness against adversarial examples by creating and incorporating adver-

*Equal contribution.
†Corresponding Author.

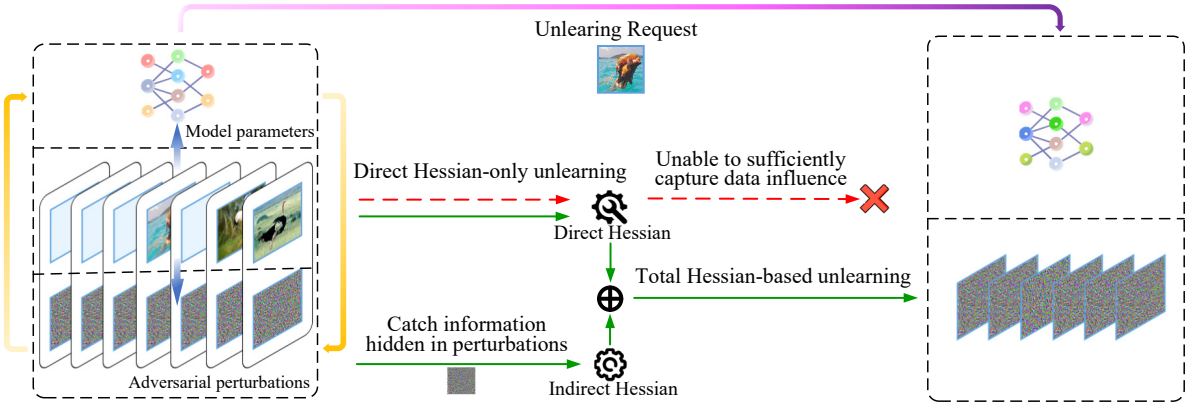

Figure 1: ATM has two interdependent sets of optimization variables, model parameters and adversarial perturbations (yellow arrows), both of which contain nested training data influence (blue arrows). To remove such nested data influence, ATM unlearning requires the total Hessian that consists of both direct and indirect Hessian components (green solid lines). Existing unlearning methods designed for standard training models are inapplicable to ATM unlearning, because they only use direct Hessian information during the unlearning update and do not capture sufficient data influence (red dashed lines).

sarial examples into the training process. ATM is trained by a bi-level optimization with the model parameters as the outer-level variable and the adversarial perturbations as the inner-level variable. To the best of our knowledge, all existing unlearning methods focus solely on data removal from STM. As our analysis will reveal in Section 3, the existing methods cannot be applied to ATM unlearning without miscapturing the data influence to be removed, due to ATM's bi-level optimization structure.

In this paper, we propose a new unlearning approach called *MUter* : **M**achine **U**nlearning for da**t**a removal from adv**e**rsarial t**r**aining models. First, we define the ATM unlearning task in accordance with the mainstream machine unlearning standard. Second, we convert it to an ATM unlearning criteria derived from the optimality condition of ATM, which facilitates the derivation of the unlearning update for ATM. Meanwhile, existing unlearning methods designed for STM cannot satisfy the ATM unlearning criteria. As illustrated in Figure 1, ATM, as a bi-level optimization, has two sets of variables: model parameters and adversarial perturbations, which are interdependent and both contain training data information. In contrast, STM, as a single-level optimization, has only one set of variables: model parameters, which have a direct influence dependence on each training datapoint. Existing unlearning methods cannot sufficiently remove data influence from ATM because they fail to account for the indirect influence on the model parameters (i.e., miss the indirect Hessian part in Figure 1). That is, due to the coupled outer-inner optimization, any updates to the outer model parameters will incur further updates for both the adversarial perturbations and the model parameters. To address this, we derive a new closed-form unlearning update for ATM, which is underpinned by the total Hessian-based influence measure. Third, based on the ATM unlearning update, we propose a three-stage unlearning framework to support successive unlearning requests for ATM, which enhances efficiency by selectively storing certain computations in memory. Last, we leverage Schur complement conversion and Neumann series approximation to avoid the computationally demanding and numerical unstable computations of Hessian matrix and its inversions.

To summarize, our main contributions are:

1. We introduce the problem of unlearning from adversarial training models, which is a new and pressing challenge to simultaneously meet emerging privacy regulations and ensure the adversarial robustness of the model. To the best of our knowledge, it has not been studied in the existing literature.
2. We derive a new unlearning update tailored to ATM unlearning, which has the total Hessian-based data influence measure to sufficiently account for the interdependence between inner and outer optimizations of ATM.
3. We propose *MUter* based on the proposed unlearning update, which supports successive unlearning requests for ATM and introduces a series of conversions and approximations for Hessian inversions to alleviate the computational cost and improve the numerical stability.
4. We perform a comprehensive evaluation to verify that *MUter* achieves effective and efficient unlearning performance while maintaining the model accuracy and adversarial robustness, under two unlearning settings on four datasets.

2. Preliminaries and Related Work

2.1. Setup and Notation

Setup. Denote the training dataset with n training samples by $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, where for $\forall i \in$

$\{1, \ldots, n\}$, $\mathbf{x}_i \in \mathbb{R}^d$ is the d-dimensional feature vector, and $y_i \in \mathbb{R}$ is the corresponding response/label. Denote the loss function by $l(\boldsymbol{\omega}, \mathbf{x}_i)$, where $\boldsymbol{\omega} \in \mathbb{R}^p$ represents the vector of the p-dimensional model parameters.

Standard Training Model (STM). Standard training model refers to the empirical risk minimization model: $\boldsymbol{\omega}^*_{std} = \text{argmin}_{\boldsymbol{\omega}} \frac{1}{n} \sum_{i=1}^n l(\boldsymbol{\omega}, \mathbf{x}_i)$, where $\boldsymbol{\omega} \in \mathbb{R}^p$ is the model parameter variable.

Adversarial Training Model (ATM). Adversarial training model is an effective defense against robustness threats posed by the adversarial attack. In this paper, we adopt one of the most widely-used ATM variants to consider the following bi-level robust optimization formulation [30, 50],

$$\boldsymbol{\omega}^* = \underset{\boldsymbol{\omega}}{\text{argmin}} \frac{1}{n} \sum_{i=1}^n \max_{\boldsymbol{\delta}_i \in \mathcal{B}(\mathbf{x}_i, r)} l(\boldsymbol{\omega}, \mathbf{x}_i + \boldsymbol{\delta}_i), \tag{1}$$

where the inner optimization is to find adversarial perturbation $\boldsymbol{\delta}_i$ from the constraint set $\mathcal{B}(\mathbf{x}_i, r)$ to maximize the loss and the outer optimization is to find the model parameter $\boldsymbol{\omega}$ that minimizes the loss for the adversarially perturbed examples. To make the interdependence between the outer variable $\boldsymbol{\omega}$ and the inner variable δ more explicit, we further introduce an auxiliary function $\delta_i(\boldsymbol{\omega})$ given by

$$\delta_i(\boldsymbol{\omega}) = \underset{\boldsymbol{\delta}_i \in \mathcal{B}(\mathbf{x}_i, r)}{\text{argmax}} \; l(\boldsymbol{\omega}, \mathbf{x}_i + \boldsymbol{\delta}_i). \tag{2}$$

Machine Unlearning for Adversarial Training Models. Definition 1 below specifies the ATM unlearning task in consistency with the mainstream unlearning standard. For notational simplicity, we focus on single-point removal in the paper, but the analysis and algorithms developed in the paper are generalizable to batch removal as well (please refer to Appendix B.5 for the batch removal generalization).

Definition 1. (Machine Unlearning for Adversarial Training Models) For a trained adversarial training model as in eq.(1), let $\forall\, i^\dagger \in \{1, \ldots, n\}$ be the index of a datapoint to be forgotten. The ATM after machine unlearning unlearning (forgetting datapoint $i^\dagger$) has parameter $\boldsymbol{\omega}^*_{-i^\dagger}$ as follows,

$$\boldsymbol{\omega}^*_{-i^\dagger} = \underset{\boldsymbol{\omega}}{\text{argmin}} \frac{1}{n-1} \sum_{i=1, i \neq i^\dagger}^n \max_{\boldsymbol{\delta}_i \in \mathcal{B}(\mathbf{x}_i, r)} l(\boldsymbol{\omega}, \mathbf{x}_i + \boldsymbol{\delta}_i). \tag{3}$$

Retraining-from-scratch serves as the golden standard and also a naive baseline to obtain the unlearned ATM $\boldsymbol{\omega}^*_{-i^\dagger}$, which is prohibitive in computation, especially for ATM with both inner maximization and outer minimization. Hence, the goal of machine unlearning for ATM is to approximately remove $i^\dagger$'s influence with great computational efficiency improvement over retraining-from-scratch.

Notation. For a twice differentiable $l(\boldsymbol{\omega}, \mathbf{x}+\boldsymbol{\delta})$, $\nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}, \mathbf{x}+\boldsymbol{\delta})$ denotes the direct gradient of l with respect to (w.r.t.) $\boldsymbol{\omega}$ and $\partial_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}, \mathbf{x}+\boldsymbol{\delta})$, $\partial_{\boldsymbol{\omega}\boldsymbol{\delta}} l(\boldsymbol{\omega}, \mathbf{x}+\boldsymbol{\delta})$, $\partial_{\boldsymbol{\delta}\boldsymbol{\omega}} l(\boldsymbol{\omega}, \mathbf{x}+\boldsymbol{\delta})$, $\partial_{\boldsymbol{\delta}\boldsymbol{\delta}} l(\boldsymbol{\omega}, \mathbf{x}+\boldsymbol{\delta})$ denote the second order partial derivatives w.r.t. $\boldsymbol{\omega}$ and $\boldsymbol{\delta}$, correspondingly. The total Hessian is

$$\mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}, \mathbf{x}+\boldsymbol{\delta}) := \overbrace{\partial_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}, \mathbf{x}+\boldsymbol{\delta})}^{\text{Direct Hessian}} - \overbrace{\partial_{\boldsymbol{\omega}\boldsymbol{\delta}} \partial^{-1}_{\boldsymbol{\delta}\boldsymbol{\delta}} \partial_{\boldsymbol{\delta}\boldsymbol{\omega}} l(\boldsymbol{\omega}, \mathbf{x}+\boldsymbol{\delta})}^{\text{Indirect Hessian}}, \tag{4}$$

where $\partial_{\boldsymbol{\omega}\boldsymbol{\delta}} \partial^{-1}_{\boldsymbol{\delta}\boldsymbol{\delta}} \partial_{\boldsymbol{\delta}\boldsymbol{\omega}} l(\boldsymbol{\omega}, \mathbf{x}+\boldsymbol{\delta}) = \partial_{\boldsymbol{\omega}\boldsymbol{\delta}} l(\boldsymbol{\omega}, \mathbf{x}+\boldsymbol{\delta})(\partial_{\boldsymbol{\delta}\boldsymbol{\delta}} l(\boldsymbol{\omega}, \mathbf{x}+\boldsymbol{\delta}))^{-1} \partial_{\boldsymbol{\delta}\boldsymbol{\omega}} l(\boldsymbol{\omega}, \mathbf{x}+\boldsymbol{\delta})$. We also use the short hand notation $l^*_i := l(\boldsymbol{\omega}^*, \mathbf{x}_i + \delta_i(\boldsymbol{\omega}^*))$ and similarly introduce $\nabla_{\boldsymbol{\omega}} l^*_i$, $\partial_{\boldsymbol{\omega}\boldsymbol{\omega}} l^*_i$, $\partial_{\boldsymbol{\omega}\boldsymbol{\delta}} l^*_i$, $\partial_{\boldsymbol{\delta}\boldsymbol{\omega}} l^*_i$, $\partial_{\boldsymbol{\delta}\boldsymbol{\delta}} l^*_i$.

2.2. Related Work

We briefly review existing machine unlearning methods. They are exclusively focused on STM and are not applicable to ATM unlearning with sufficient data influence removal. We provide a brief overview of current machine unlearning methods, which are focused exclusively on unlearning for STM and are not applicable to ATM unlearning with sufficient data influence removal. Thus, it is necessary to develop a new unlearning approach specific to ATM unlearning.

Exact Machine Unlearning. Exact unlearning methods perform exact retraining but manage to do so only on a selective portion of the dataset to avoid retraining on the entire dataset. Some exact unlearning methods are designed for specific models like Naive Bayes [8], quantized k-means [19], and random forests [6]. Recently, SISA [5] proposes a more general exact unlearning strategy. It divides the training data into multiple disjoint shards during the training phase, and retrains only on the shard that contains the data to be removed during the unlearning phase. Later, [11] and [12] extend the SISA strategy to unlearning for recommendation systems and graph neural networks. However, the SISA strategy requires different shards to be independently updatable from each other, which is not the case for ATM due to its interdependent bi-level structure.

Approximate Machine Unlearning. Approximate unlearning methods seek the updated model parameters to approximately satisfy the optimality condition of the objective function on the remaining data, which can be roughly divided into three categories. 1) Unlearning with direct Hessian-related terms: [22, 26, 31, 34, 40, 43, 49] propose Newton step-based unlearning updates, where the Newton curvature matrices are direct Hessian-related terms. These methods also inject Gaussian noise to further destroy the remaining data influence due to the discrepancy between the exact unlearning standard and the approximate unlearning criteria. Some works also consider to alleviate the computational cost of the Hessian. For example, [22] proposes to approximate the direct Hessian with Fisher information matrix; [43] proposes to avoid dealing with the entire direct Hessian by selecting small Hessian blocks for unlearning update. 2) Unlearning based on neural tangent kernel (NTK) theory: they approximates the training phase by NTK theory, where [23] regards

the training as an approximately linear change and [21] divides the training phase into two stages, one is the linear standard training on the core dataset, and the other is the linear finetuning on the target dataset. Both works then applies direct Hessian-related unlearning updates. 3) Unlearning by tracking the Stochastic Gradient Descent (SGD) path during training: DeltaGrad [60] and unrollSGD [57] propose to track and reverse the SGD optimization path during the training phase. However, all above methods do not consider the interdependence between model parameters and adversarial perturbations of ATM, which cannot holistically capture the data influence, as will be detailed in the next section.

3. Unlearning Update for ATM

ATM Unlearning Criteria. We begin by converting eq.(3) in Definition 1 to the optimality condition form of ATM based on Danskin's Theorem [13],

$$\sum_{i=1,i\neq i^\dagger}^{n} \nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*_{-i^\dagger}, \mathbf{x}_i + \delta_i(\boldsymbol{\omega}^*_{-i^\dagger})) = 0, \tag{5}$$

which provides more convenience to the follow-up ATM unlearning designs. Our goal is to find a closed-form unlearning update $\mathcal{U}(\boldsymbol{\omega}^*, i^\dagger)$ such that the unlearning model with parameter $\boldsymbol{\omega}^u = \boldsymbol{\omega}^* + \mathcal{U}(\boldsymbol{\omega}^*, i^\dagger)$ will satisfy the following **ATM unlearning criteria**,

$$\sum_{i=1,i\neq i^\dagger}^{n} \nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^u, \mathbf{x}_i + \delta_i(\boldsymbol{\omega}^u)) \approx 0. \tag{6}$$

Cause of Inapplicability of Existing Approximate Unlearning Methods. Before deriving our ATM unlearning update, we recall the design rationale common to existing approximate machine unlearning methods and point out why they fail to sufficiently remove the data influence for ATM. Despite the varying forms of specific unlearning updates, existing approximate unlearning methods all exploit the trained model parameter $\boldsymbol{\omega}^*$ and connect it with the unlearned model $\boldsymbol{\omega}^u$ by first-order expansion of $\nabla_{\boldsymbol{\omega}} l$ around $\boldsymbol{\omega}^*$:

$$\begin{aligned}
&\frac{1}{n-1}\sum_{i=1,i\neq i^\dagger}^{n} \nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^u, \mathbf{x}_i + \delta_i(\boldsymbol{\omega}^u)) \\
&\approx \frac{1}{n-1}\sum_{i=1,i\neq i^\dagger}^{n} \big[\nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i + \delta_i(\boldsymbol{\omega}^u)) \\
&+ \partial_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i + \delta_i(\boldsymbol{\omega}^u))(\boldsymbol{\omega}^u - \boldsymbol{\omega}^*)\big].
\end{aligned} \tag{7}$$

Existing methods then obtain the updated $\boldsymbol{\omega}^u$ by letting the right hand side of eq.(7) approach 0 and exploiting the optimality condition of $\boldsymbol{\omega}^*$. For example, the influence function-inspired unlearning [26, 49] has $\mathcal{U}(\boldsymbol{\omega}^*, i^\dagger) =$

$$\Big[\sum_{i=1,i\neq i^\dagger}^{n} \underbrace{\partial_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i + \delta_i(\boldsymbol{\omega}^u))}_{\text{Direct Hessian-only}}\Big]^{-1} \nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_{i^\dagger} + \delta_{i^\dagger}(\boldsymbol{\omega}^u)). \tag{8}$$

However, there are two obvious issues when applying the above design rationale to the ATM unlearning: 1) the conversion from eq.(7) to eq.(8) requires $\sum_{i=1}^{n} \nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i + \delta_i(\boldsymbol{\omega}^u)) = 0$, which does not hold in general. Rather, the optimality condition only promises $\sum_{i=1}^{n} \nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i + \delta_i(\boldsymbol{\omega}^*)) = 0$; 2) the Hessian $\partial_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i + \delta_i(\boldsymbol{\omega}^u))$ and gradient $\nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i + \delta_i(\boldsymbol{\omega}^u))$ in eq.(8) are not commutable without knowing $\boldsymbol{\omega}^u$ in advance. We point out that *the root cause of both issues is that eq.(7) fails to account for the interdependence between $\boldsymbol{\omega}$ and $\boldsymbol{\delta}$ as in eq.(2).*

Proposed ATM Unlearning Update. The above root cause propels us to derive a new unlearning update that can holistically capture the nested data influence. To account for the interdependence between the adversarial perturbations and the model parameters in eq.(2), we take the complete expansion w.r.t. both $\boldsymbol{\omega}$ and $\boldsymbol{\delta}$ as in Lemma 1 below.

Lemma 1. *By expansion around both $\boldsymbol{\omega}^*$ and $\delta(\boldsymbol{\omega}^*)$, the approximate machine unlearning model parameter $\boldsymbol{\omega}^u$ and the original model parameter $\boldsymbol{\omega}^*$ has the following relation,*

$$\begin{aligned}
&\frac{1}{n-1}\sum_{i=1,i\neq i^\dagger}^{n} \nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^u, \mathbf{x}_i + \delta_i(\boldsymbol{\omega}^u)) \\
\approx&\frac{1}{n-1}\sum_{i=1,i\neq i^\dagger}^{n} \Big[\nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i + \delta_i(\boldsymbol{\omega}^*)) \\
&+ \partial_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i + \delta_i(\boldsymbol{\omega}^*))(\boldsymbol{\omega}^u - \boldsymbol{\omega}^*) \\
&+ \partial_{\boldsymbol{\omega}\boldsymbol{\delta}} l(\boldsymbol{\omega}^*, \mathbf{x}_i + \delta_i(\boldsymbol{\omega}^*))(\delta_i(\boldsymbol{\omega}^u) - \delta_i(\boldsymbol{\omega}^*))\Big].
\end{aligned} \tag{9}$$

Proof. Please refer to Appendix B.1. □

The last line of eq.(9) is key to the sufficient removal of the data influence from ATM, which captures the interdependence between the adversarial perturbations and the model parameters. Based on Lemma 1, we can derive the ATM unlearning update in closed-form in Theorem 1 below.

Theorem 1. *Consider the adversarial training model with trained model parameter $\boldsymbol{\omega}^*$ as in eq.(1). Let $(\mathbf{x}_{i^\dagger}, y_{i^\dagger})$ be a datapoint in the training dataset to be forgotten. Let the machine learning update $\mathcal{U}(\boldsymbol{\omega}^*, i^\dagger)$ take the following form*

$$\begin{aligned}
\mathcal{U}(\boldsymbol{\omega}^*, i^\dagger) :=& \Big[\sum_{i=1,i\neq i^\dagger}^{n} \underbrace{\mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i + \delta_i(\boldsymbol{\omega}^*))}_{\textit{Total Hessian}}\Big]^{-1} \\
&\cdot \Big[\nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_{i^\dagger} + \delta_{i^\dagger}(\boldsymbol{\omega}^*))\Big].
\end{aligned} \tag{10}$$

Then, the unlearning model with updated parameters $\boldsymbol{\omega}^u := \boldsymbol{\omega}^* + \mathcal{U}(\boldsymbol{\omega}^*, i^\dagger)$ satisfies the approximate unlearning criteria for the adversarial training model in eq.(6).

Proof. Please refer to Appendix B.1. □

It turns out that the ATM unlearning in eq.(10) requires both direct Hessian and indirect Hessian (i.e., total Hessian

in eq.(4)) to sufficiently capture the data influence, while previous methods for the STM unlearning have only the direct Hessian-related part (e.g., eq.(8)). The total Hessian term also appears in the second-order bi-level optimization literature, e.g., the Complete Newton method in [64].

A New Influence Function. Finally, we would like to remark that eq.(10) can be used to define a new influence function as follows to measure the nested data influence of a data point for ATM. It could serve as an independent interest, for example, to measure the marginal contribution of a data point to the model for valuation in data market [46] and influence-based defense against data poisoning attacks [18].

Definition 2. (Leave-one-out Adversarial Influence Function) For the adversarial training model given in eq.(1), the leave-one-out adversarial influence function for any $(\mathbf{x}, y)$ in the training dataset is defined as

$$\mathcal{I}(\mathbf{x}, y) = -\left[\frac{1}{n}\sum_{i=1}^{n} \mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i + \boldsymbol{\delta}_i^*)\right]^{-1} \nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x} + \boldsymbol{\delta}^*).$$

Definition 2 utilizes total Hessian instead of direct Hessian-only as in [32], which captures the interdependence between adversarial perturbations and model parameters. This is different from recent work [14] which defines the adversarial influence function as a training sample's influence difference between *two models*: an STM and an ATM, while Definition 2 measures a training sample's influence difference when it is in or out for a *single* ATM.

4. Proposed Method: *MUter*

Equipped with the ATM unlearning update, we propose *MUter*, a three-stage unlearning method for ATM, which has two design considerations: 1) Support the more practical successive unlearning setting where multiple data points can be forgotten in sequence by keeping a selective variable set in memory (Sec.4.1); 2) Reduce the per-unlearning computational cost by introducing more efficient approximations and conversions to avoid direct computations of the most computational demanding Hessian inversions (Sec.4.2). The overall framework of *MUter* is illustrated in Figure 2. We also present the complete algorithm in Appendix B.4.

4.1. Three Stages of *MUter*

Successive Unlearning Setting. Denote the indices of datapoints that have already been forgotten at timestamp r by $\{i_1^\dagger, \ldots, i_r^\dagger\}$, the index of datapoint to be forgotten at timestamp $r+1$ by $i_{r+1}^\dagger = i^\dagger$. Corollary 1 below extends Theorem 1 to support successive unlearning for ATM.

Corollary 1. *Considering the successive unlearning setting, let the machine unlearning update at the $r+1$-th timestamp $\mathcal{U}_r(\boldsymbol{\omega}^*, i_{r+1}^\dagger)$ take the following form*

$$\mathcal{U}_r(\boldsymbol{\omega}^*, i_{r+1}^\dagger) := \Bigg\{ \overbrace{\Big[\sum_{i=1}^{n} \mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i + \boldsymbol{\delta}_i^*)\Big]}^{\mathcal{M}_0[\mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}}]} - \overbrace{\Big[\sum_{i=i_1^\dagger}^{i_r^\dagger} \mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i + \boldsymbol{\delta}_i^*)\Big]}^{\text{Part of } \mathcal{M}_r[\mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}}]} - \Big[\mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_{i^\dagger} + \boldsymbol{\delta}_{i^\dagger}^*)\Big] \Bigg\}^{-1} \cdot \overbrace{\Big[\sum_{i=i_1^\dagger}^{i_r^\dagger} \nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i + \boldsymbol{\delta}_i^*) + \nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_{i^\dagger} + \boldsymbol{\delta}_{i^\dagger}^*)\Big]}^{\mathcal{M}_r[\nabla_{\boldsymbol{\omega}}]}. \quad (11)$$

Then, the unlearning model with updated parameters $\boldsymbol{\omega}_{r+1}^u := \boldsymbol{\omega}^* + \mathcal{U}_r(\boldsymbol{\omega}^*, i_{r+1}^\dagger)$ satisfies the approximate unlearning criteria for the adversarial training model in eq.(6).

Proof. Please refer to Appendix B.2. □

Stage I. Pre-Unlearning. According to Corollary 1, it is a natural idea to pre-compute and keep in memory the total Hessian $\left[\sum_{i=1}^{n} \mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i + \boldsymbol{\delta}_i^*)\right]$ once the ATM is trained and before responding to unlearning requests, since it does not depend on the new point to be removed. Thus, we initialize a memory set $\mathcal{M}_0$ at $r = 0$, which has three memory parts: $\mathcal{M}_0[\mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}}]$ for $\left[\sum_{i=1}^{n} \mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i + \boldsymbol{\delta}_i^*)\right]$, $\mathcal{M}_0[\boldsymbol{\omega}^*]$ for $\boldsymbol{\omega}^*$, and $\mathcal{M}_0[\nabla_{\boldsymbol{\omega}}]$ for $\nabla_{\boldsymbol{\omega}} = \mathbf{0}$ at $r = 0$.

Stage II. Unlearning. Upon receiving a new unlearning request to forget the $i^\dagger$-th datapoint at timestamp $r+1$, Corollary 1 indicates that it suffices to compute the total Hessian and gradient only on $i^\dagger$ rather than the entire training set, once the memory is set up and maintained. We evaluate the following three parts in sequence: 1) The gradient on $i^\dagger$: $\nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_{i^\dagger} + \boldsymbol{\delta}_{i^\dagger}^*)$; 2) The total Hessian on $i^\dagger$: $\mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_{i^\dagger} + \boldsymbol{\delta}_{i^\dagger}^*)$; 3) The unlearning update $\mathcal{U}_r(\boldsymbol{\omega}^*, i_{r+1}^\dagger)$ in eq.(11) and the updated parameters $\boldsymbol{\omega}_{r+1}^u$. In addition, we can also inject Gaussian noise by $\boldsymbol{\omega}_{r+1}^u + \mathbf{n}$ with $\mathbf{n} \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$ to further smooth the remaining data influence to the discrepancy between eq.(3) and eq.(6).

Stage III. Post-Unlearning. To support subsequent unlearning requests, we update $\mathcal{M}_{r+1}$ based on Corollary 1:

$$\mathcal{M}_{r+1}[\mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}}] = \mathcal{M}_r[\mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}}] - \mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_{i^\dagger} + \boldsymbol{\delta}_{i^\dagger}^*); \quad (12)$$

$$\mathcal{M}_{r+1}[\nabla_{\boldsymbol{\omega}}] = \mathcal{M}_r[\nabla_{\boldsymbol{\omega}}] + \nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_{i^\dagger} + \boldsymbol{\delta}_{i^\dagger}^*), \quad (13)$$

where the former is the online update for $\left[\sum_{i=1}^{n} \mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i + \boldsymbol{\delta}_i^*)\right] - \left[\sum_{i=i_1^\dagger}^{i_{r+1}^\dagger} \mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i + \boldsymbol{\delta}_i^*)\right]$ and the latter is for $\left[\sum_{i=i_1^\dagger}^{i_{r+1}^\dagger} \nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i + \boldsymbol{\delta}_i^*)\right]$.

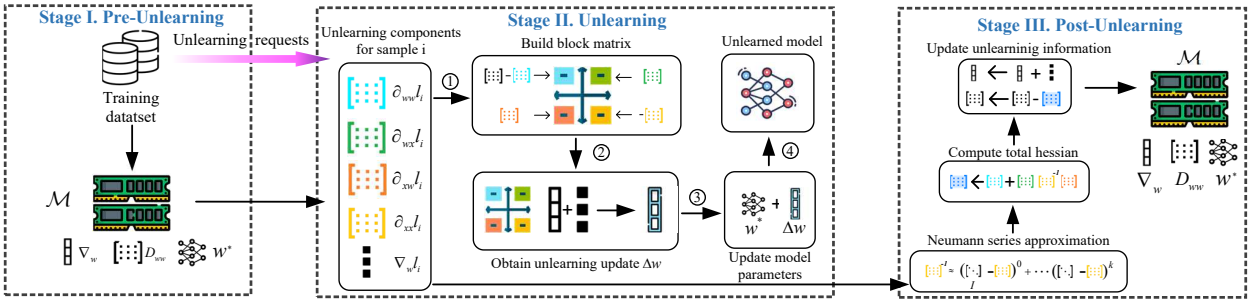

Figure 2: The proposed *MUter* framework. Stage I: the pre-unlearning stage prepares the initial memory $\mathcal{M}_0[\mathtt{D}_{\omega\omega}], \mathcal{M}_0[\nabla_\omega], \mathcal{M}_0[\boldsymbol{\omega}^*]$. Stage II: the unlearning stage computes the total Hessian and gradient on the data to be forgotten and computes the ATM unlearning update by solving a linear system. Stage II: the post-unlearning stage updates $\mathcal{M}_{r+1}[\mathtt{D}_{\omega\omega}]$ and $\mathcal{M}_{r+1}[\nabla_\omega]$ to be prepared for the next unlearning request.

4.2. Unlearning without Direct Hessian Inversions

We now present the approximations and conversions to circumvent the most computationally expensive and potentially numerical unstable parts in the three stages, which are three Hessian matrix inversions: (Inv-1) The total Hessian inversion of $\left[\mathcal{M}_r[\mathtt{D}_{\omega\omega}] - \mathtt{D}_{\omega\omega} l(\boldsymbol{\omega}^*, \mathbf{x}_{i^\dagger} + \boldsymbol{\delta}^*_{i^\dagger})\right]^{-1}$; (Inv-2) The partial Hessian inversion of $\partial^{-1}_{\boldsymbol{\delta}\boldsymbol{\delta}} l(\boldsymbol{\omega}^*, \mathbf{x}_{i^\dagger} + \boldsymbol{\delta}^*_{i^\dagger})$ inside the total Hessian in Stage II; (Inv-3) The same $\partial^{-1}_{\boldsymbol{\delta}\boldsymbol{\delta}} l(\boldsymbol{\omega}^*, \mathbf{x}_{i^\dagger} + \boldsymbol{\delta}^*_{i^\dagger})$ but in Stage I & III.

Avoid Matrix Inversions (Inv-1/2) jointly by Schur Complement Conversion. After expanding the total Hessian as in eq.(4), we can reorganize $\left[\mathcal{M}_r[\mathtt{D}_{\omega\omega}] - \mathtt{D}_{\omega\omega} l(\boldsymbol{\omega}^*, \mathbf{x}_{i^\dagger} + \boldsymbol{\delta}^*_{i^\dagger})\right]$ into the following form $\mathbf{S} = \left[\mathcal{M}_r[\mathtt{D}_{\omega\omega}] - \partial_{\omega\omega} l^*_{i^\dagger}\right] - \left[-\partial_{\omega\boldsymbol{\delta}} \partial^{-1}_{\boldsymbol{\delta}\boldsymbol{\delta}} \partial_{\boldsymbol{\delta}\omega} l^*_{i^\dagger}\right]$, which can be regarded as the Schur complement of the block matrix $\mathbf{H}$:

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_{11} & \mathbf{H}_{12} \\ \mathbf{H}_{21} & \mathbf{H}_{22} \end{bmatrix} = \begin{bmatrix} \mathcal{M}_r[\mathtt{D}_{\omega\omega}] - \partial_{\omega\omega} l^*_{i^\dagger} & \partial_{\omega\boldsymbol{\delta}} l^*_{i^\dagger} \\ \partial_{\boldsymbol{\delta}\omega} l^*_{i^\dagger} & -\partial_{\boldsymbol{\delta}\boldsymbol{\delta}} l^*_{i^\dagger} \end{bmatrix}. \tag{14}$$

By Schur complement lemma deferred to Appendix B.3 for completeness, we convert the inversion of $\mathbf{S}$ to the inversion of $\mathbf{H}$, then the unlearning update is equivalent to solving a linear system, as summarized in the following Theorem.

Theorem 2. *The unlearning update $\mathfrak{U}_r(\boldsymbol{\omega}^*, i^\dagger_{r+1})$ in eq.(11) can be equivalently computed by solving the linear system:*

$$\begin{bmatrix} \mathcal{M}_r[\mathtt{D}_{\omega\omega}] - \partial_{\omega\omega} l^*_{i^\dagger} & \partial_{\omega\mathbf{x}} l^*_{i^\dagger} \\ \partial_{\mathbf{x}\omega} l^*_{i^\dagger} & -\partial_{\mathbf{x}\mathbf{x}} l^*_{i^\dagger} \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{\omega} \\ \Delta\boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} \mathcal{M}_r[\nabla_\omega] + \nabla_\omega l^*_{i^\dagger} \\ \mathbf{0} \end{bmatrix},$$

where $\Delta\boldsymbol{\omega}$ is the unlearning update $\mathfrak{U}_r(\boldsymbol{\omega}^*, i^\dagger_{r+1})$ and $\Delta\boldsymbol{\alpha}$ is an auxiliary variable that can be discarded.

The linear system in Theorem 2 can be solved by conjugate gradient or fixed point methods, which are more efficient than computing the Hessian inversions (Inv-1/2) directly.

Avoid Matrix Inversions (Inv-3) by Neumann Series Approximation. We expand $\partial^{-1}_{\boldsymbol{\delta}\boldsymbol{\delta}} l^*_i$ by Neumann Series: $\partial^{-1}_{\boldsymbol{\delta}\boldsymbol{\delta}} l^*_i = \lim_{k\to\infty} \sum_{j=0}^{k} \left[\mathbf{I} - \partial_{\mathbf{x}\mathbf{x}} l^*_i\right]^j$. Then, clipping at order k, we have the following approximation,

$$\partial^{-1}_{\boldsymbol{\delta}\boldsymbol{\delta}} l^*_i \approx \mathbf{I} + \left[\mathbf{I} - \partial_{\boldsymbol{\delta}\boldsymbol{\delta}} l^*_i\right] + \cdots + \left[\mathbf{I} - \partial_{\boldsymbol{\delta}\boldsymbol{\delta}} l^*_i\right]^k. \tag{15}$$

Each term in memory $\mathcal{M}_r[\mathtt{D}_{\omega\omega}]$ used in Stage I & III can be approximated by $\mathtt{D}_{\omega\omega} l^*_i \approx \widetilde{\mathtt{D}_{\omega\omega} l^*_i} :=$

$$\partial_{\omega\omega} l^*_i - \partial_{\omega\boldsymbol{\delta}} l^*_i \Big(\sum_{j=0}^{k} \left[\mathbf{I} - \partial_{\boldsymbol{\delta}\boldsymbol{\delta}} l^*_i\right]^j \Big) \partial_{\boldsymbol{\delta}\omega} l^*_i. \tag{16}$$

5. Experiments

We conduct experimental evaluations on two groups of machine learning models and four common datasets, under two typical experiment settings in the unlearning literature [21–23, 26]. Our source code, experiment details, and more experiment results can be found in Supplement.

5.1. Experiment Setup

5.1.1 Models and Datasets

Linear Models. We consider two linear models: 1) Ridge Regression (RR) with least square loss; 2) Logistic Regression (LR) with logistic loss, both are regularized by squared ℓ_2-norm with hyperparameter $\lambda = 1e-4$.

Neural Network Models. We utilize the Wide ResNet model (i.e., Wide ResNet 28-10 model) [63] and consider the pretraining setting [22, 26] (also similar to the mixed-privacy removal setting [21]) for the neural network model as follows. We first pretrain a model on a *core dataset* with adversarial training. We then finetune the model on another *target dataset* with adversarial finetuning [28], where all layers are frozen except the last layer. We consider unlearning requests from only the target dataset used for the adversarial finetuning.

Adversarial Training/Finetuning Algorithms. For outer-level, we utilize SGD to optimize the model parameters $\boldsymbol{\omega}^*$; for inner-level, we utilize both PGD [39] and FGSM [24] to generate adversarial perturbations. The experiment results by PGD are reported in the paper, while the results by FGSM and more detailed training and finetuning settings are relegated to Appendix A.1 and A.3.

Datasets. We consider four common datasets for the two settings above: *(linear models)* **i) MNIST-b** is the subset from MNIST with classes '1' and '7' for binary classification purpose; **ii) Covtype** is a dataset from the LIBSVM repository [10]; *(neural network model with pretraining)* **iii) CIFAR-10** (target dataset) and the Downsampled ImageNet (core dataset) are both natural image datasets; **iv) Lacuna-10** (target dataset) and Lacuna-100 (core dataset) are both face datasets. More detail can be found in Appendix A.2.

5.1.2 Baseline Unlearning Methods

We compare *MUter* with six approximate unlearning methods (with their abbreviations in bold), all of which are originally designed for unlearning from standard training models. Since they utilize different Hessian terms in the unlearning update, for a fair comparison, we evaluate them under the same three-stage framework as *MUter* but substitute in their corresponding unlearning update formulations. (1) **Newton** unlearning [26] and (2) Newton unlearning measured with adversarial perturbation (**Newton-delta**): both methods utilize the Newton step as the unlearning update, where the former computes at the original samples (i.e., $\left[\sum_{i=1,i\neq i^\dagger}^{n} \partial_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i)\right]^{-1} \nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_{i^\dagger})$) and the latter at the adversarially perturbed samples (i.e., $\left[\sum_{i=1,i\neq i^\dagger}^{n} \partial_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i + \boldsymbol{\delta}_i^*)\right]^{-1} \nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_{i^\dagger} + \boldsymbol{\delta}_{i^\dagger}^*)$). Similarly, we have (3) **Fisher** unlearning [22] and (4) **Fisher-delta**, which utilize the Fisher matrix $(\nabla_{\boldsymbol{\omega}} l \cdot \nabla_{\boldsymbol{\omega}}^\top l)$ to approximate the direct Hessian $\partial_{\boldsymbol{\omega}\boldsymbol{\omega}} l$. (5) **Influence** function-based unlearning [32, 40] and (6) **Influence-delta**, which utilize the influence function for STM for the unlearning update.

Meanwhile, we utilize (7) **Retrain**-from-scratch as the golden standard reference, which applies the same adversarial training/finetuning algorithms to retrain/re-finetune the model on the remaining data to obtain $\boldsymbol{\omega}_{-i^\dagger}^*$ in eq.(3).

5.1.3 Evaluation Metrics

We measure four aspects of the unlearning performance: 1) **Effectiveness** measures the closeness of the unlearned ATM compared to the golden standard **Retrain**, for which we utilize the ℓ_2-norm difference between the model parameter vectors, i.e., $\|\boldsymbol{\omega}^u - \boldsymbol{\omega}_{-i^\dagger}^*\|_2$; 2) **Accuracy** measures the clean accuracy of the unlearned ATM, for which we utilize the accuracy on *clean test samples*; 3) **Robustness** measures the adversarial accuracy of the unlearned ATM, for which we utilize the accuracy on *adversarial perturbed test samples*; 4) **Efficiency** measures the unlearning time the unlearning method takes to respond the unlearning request, for which we report CPU time.

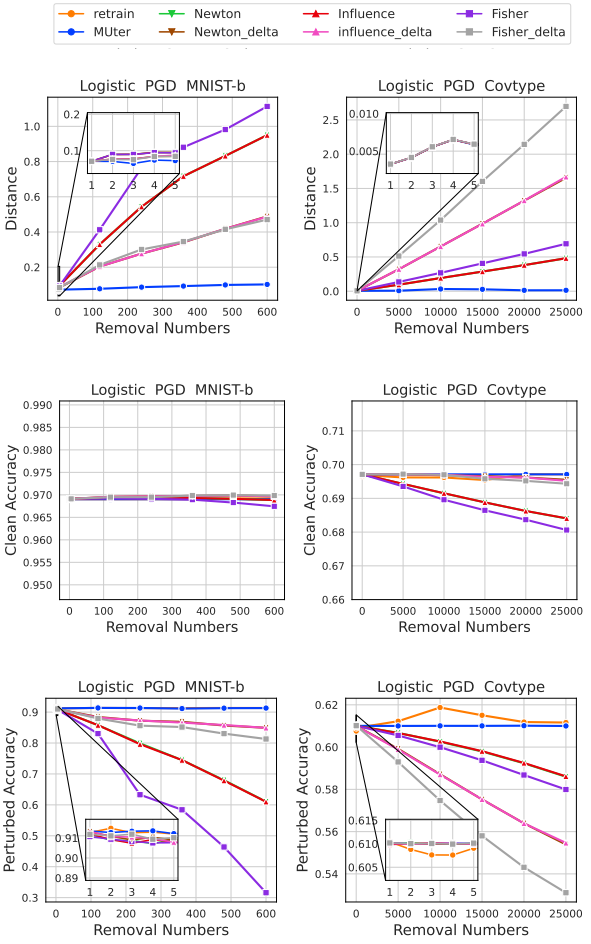

Figure 3: Evaluation results on Logistic Regression Model: Effectiveness (top), Accuracy (middle), and Robustness (bottom) on datasets MNIST-b (left column), Covtype (right column). Large plots have greater removal numbers: $1\%, 2\%, \cdots, 5\%$; the inside small plots have fewer removal numbers: $1, 2, \cdots, 5$.

5.2. Experiment Results

5.2.1 Results with Linear Model

Figure 3 reports the experiment results with logistic regression model on the MNIST-b and Covtypes datasets. More results with ridge regression are deferred to Appendix A.3.

Effectiveness. Top row of Figure 3 shows the effectiveness comparison. All compared methods have increased model parameter distance with the increasing number of unlearning

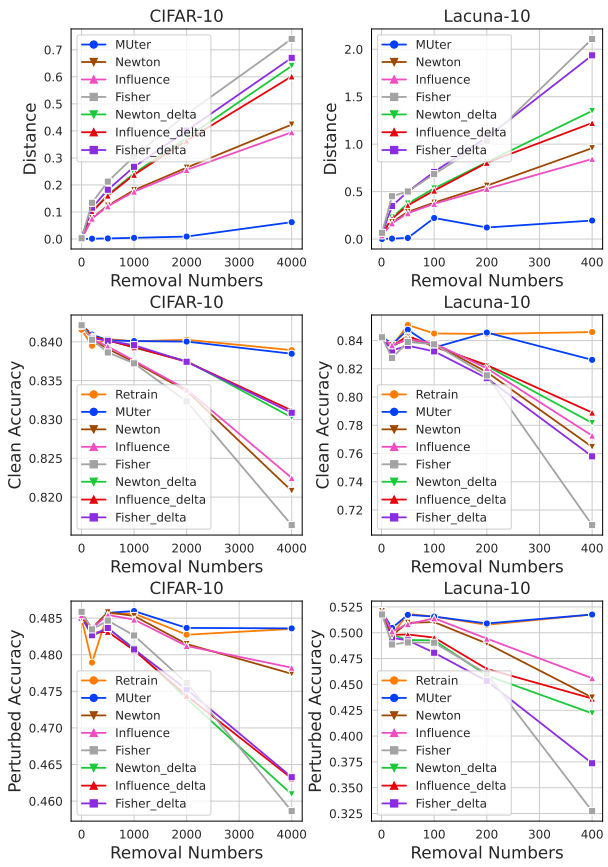

Figure 4: Evaluation results on Neural Network: Effectiveness (top), Accuracy (middle), and Robustness (bottom) on datasets Lacuna-10 (left column) and CIFAR-10 (right column), under removal numbers $(1, 0.4\%, 1\%, 2\%, 4\%, 8\%)$.

Model Type	Removal Number	MNIST-b Fisher	MNIST-b F-delta	MNIST-b *MUter*	MNIST-b Retrain	Covtype Fisher	Covtype F-delta	Covtype *MUter*	Covtype Retrain
LR	1	0.002	0.002	0.009	63	0.002	0.002	0.007	192
	5	0.010	0.012	0.052	63	0.009	0.011	0.037	194
	10	0.019	0.021	0.106	62	0.018	0.022	0.071	194
RR	1	0.002	0.002	0.009	64	0.002	0.002	0.006	200
	5	0.009	0.011	0.057	65	0.011	0.012	0.032	206
	10	0.018	0.019	0.097	65	0.019	0.020	0.064	205

Table 1: Efficiency results with linear models of logistic regression (top) and ridge regression (bottom): The unlearning time (in seconds) of **Fisher**, **Fisher-delta**, *MUter* and **Retrain** under varying removal numbers: 1, 5, 10.

requests, which is due to the decreased approximation capability of the approximate unlearning criteria. *MUter* achieves the smallest deviation from **Retrain** and outperforms all the compared methods, which is because its ATM unlearning update sufficiently captures the nested data influence in ATM.

Accuracy. Middle row of Figure 3 shows the clean accuracy comparison. All unlearning methods do not show a significant drop in clean accuracy after data forgotten. In general, *MUter* is among the methods that achieve higher accuracy and exhibits close consistency with **Retrain**.

Removal Number	Lacuna-10 Fisher	Lacuna-10 F-delta	Lacuna-10 *MUter*	Lacuna-10 Retrain	CIFAR-10 Fisher	CIFAR-10 F-delta	CIFAR-10 *MUter*	CIFAR-10 Retrain
1	1.47	1.66	4.18	935	1.51	1.73	3.81	5224
5	7.76	8.41	21.37	934	7.54	8.58	18.64	5294
10	15.51	16.82	41.69	932	15.34	16.42	37.57	5260

Table 2: Efficiency results with Neural Network: The unlearning time (in seconds) of **Fisher**, **Fisher-delta**, *MUter* and **Retrain** under varying removal numbers: 1, 5, 10.

Robustness. Bottom row of Figure 3 shows the robustness comparison. When the number of forgotten requests is small, most methods have little variation in perturbed accuracy. When the forgotten number becomes larger, all baseline unlearning methods have an obvious decrease in perturbed accuracy, while *MUter* still shows high perturbed accuracy and has the closest proximity to **Retrain**.

Efficiency. Table 1 shows the efficiency comparison. *MUter* has significant efficiency improvement over **Retrain**, which is the utmost desideratum of the unlearning method. In addition, we compare with **Fisher** and **Fisher-delta**, which are the most efficient methods among the six baseline methods due to the more efficient Fisher information matrix approximation. *MUter* takes more CPU time because it computes the total Hessian to obtain a more holistic data influence measure, while the baseline methods compute only the (approximate) direct Hessian and omit the indirect Hessian. Thus, the small increase in CPU time is the necessary tax to pay in exchange for more effective unlearning.

Tradeoff between deletion effectiveness, accuracy, and robustness. According to the above results, there is a tradeoff between efficient and deletion effectiveness, accuracy, and robustness. That is, *MUter* takes slightly longer than direct Hessian-only unlearning methods due to the additional computation of the indirect Hessian. However, this extra computation is worthwhile as *MUter* offers improved deletion effectiveness, accuracy, and robustness.

5.2.2 Results with Neural Network Model

Figure 4 summarizes the results of effectiveness, accuracy, and robustness on the two datasets for the neural network model with pretraining. *MUter* has the smallest difference of parameter distance with **Retrain**, which indicates that *MUter* generates the most similar unlearning model to the **Retrain** model. In addition, *MUter* has the most consistent behaviour with **Retrain** in terms of clean accuracy and perturbation accuracy. In terms of efficiency, Table 2 reports the comparison with **Retrain** and two most efficient baseline methods **Fisher** and **Fisher-delta**. Similar to the linear

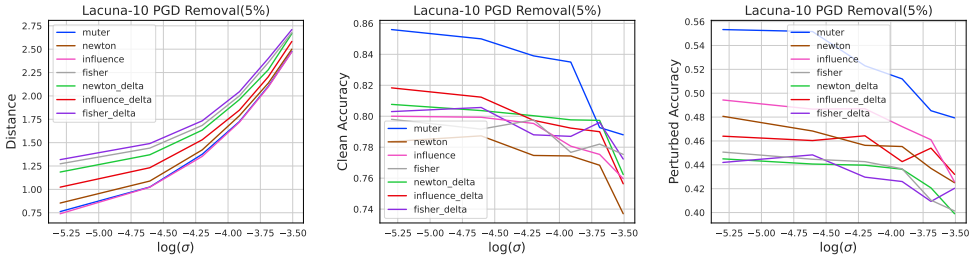

Figure 5: Effect of Gaussian noise injection: Effectiveness (left), Accuracy (middle), and Robustness (right) versus the standard variation σ of Gaussian noise in log scale, under the case of removing 5% samples and on the Lacuna-10 dataset.

model case, *MUter* has significant efficiency improvement over **Retrain**. Although *MUter* costs more unlearning time than the approximate methods, it provides significantly more effective, more accurate, and more robust unlearned ATM.

Effect of Varying Magnitudes of Gaussian Noise. Figure 5 shows the effectiveness, accuracy, and robustness comparisons under varying standard variations of Gaussian noise (σ) injected by $\boldsymbol{\omega}^u_{r+1} + \mathbf{n}$ with $\mathbf{n} \sim \mathcal{N}(0, \sigma^2\mathbf{I})$. All methods get worse unlearning model performance with larger σ, but *MUter* maintains the best performance for all σ's among all compared unlearning methods.

6. Conclusion

In this paper, we studied a new joint privacy-robustness problem of machine unlearning from adversarial training models to simultaneously meet the emerging privacy regulations on Right to be Forgotten and ensure the adversarial robustness of the model. We proposed a new unlearning method called *Muter*, which is underpinned by a total Hessian-based measure to sufficiently capture the data influence on both model parameters and adversarial perturbations. We further introduced the Schur complement conversion and the Neumann series approximation to mitigate the computational cost. Our methods show significant enhancement in effectiveness and efficiency compared to baseline methods.

As future works, several directions can be further explored: 1) extending *MUter* to other adversarial training variants like adversarial regularization; 2) further reducing the memory cost by considering low-rank/k-fact approximations to approximate the Hessian matrix, e.g., [25]; 3) studying the approximation capability of the new influence function inspired by *MUter* , e.g., across different depths of neural networks [4]; 4) exploring the potential connection between the new influence function with proximal Bregman response function [2].

7. Acknowledgement

The authors would like to thank the anonymous reviewers for their constructive comments. This research has been funded in part by National Science Foundation of China (NSFC) 62206207, 62102300, 62072395, and U20A20178, National Key Research and Development Program of China 2020AAA0107705, National Science Foundation (NSF) CNS-2124104, CNS-2125530, and National Institute of Health (NIH) R01ES033241.

References

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016. 1

[2] Juhan Bae, Nathan Ng, Alston Lo, Marzyeh Ghassemi, and Roger B. Grosse. If influence functions are the answer, then what is the question? In *NeurIPS*, 2022. 9

[3] Tao Bai, Jinqi Luo, Jun Zhao, Bihan Wen, and Qian Wang. Recent advances in adversarial training for adversarial robustness. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pages 4312–4321, 8 2021. Survey Track. 1

[4] Samyadeep Basu, Phillip Pope, and Soheil Feizi. Influence functions in deep learning are fragile. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021. 9

[5] Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *42nd IEEE Symposium on Security and Privacy*, 2021. 1, 3

[6] Jonathan Brophy and Daniel Lowd. Machine unlearning for random forests. In *International Conference on Machine Learning*, pages 1092–1104, 2021. 3

[7] Qiong Cao, Li Shen, Weidi Xie, Omkar M. Parkhi, and Andrew Zisserman. Vggface2: A dataset for recognising faces across pose and age. In *2018 13th IEEE International Conference on Automatic Face Gesture Recognition (FG 2018)*, pages 67–74, 2018. 12

[8] Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *2015 IEEE Symposium on Security and Privacy*, pages 463–480. IEEE, 2015. 1, 3

[9] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2017. 1

[10] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3), may 2011. 7

[11] Chong Chen, Fei Sun, Min Zhang, and Bolin Ding. Recommendation unlearning. In *Proceedings of the ACM Web Conference 2022*, pages 2768–2777, 2022. 3

[12] Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. Graph unlearning. In *Proceedings of the ACM Conference on Computer and Communications Security 2022*, 2022. 3

[13] John M Danskin. *The theory of max-min and its application to weapons allocation problems*, volume 5. Springer Science & Business Media, 2012. 4

[14] Zhun Deng, Cynthia Dwork, Jialiang Wang, and Linjun Zhang. Interpreting robust optimization via adversarial influence functions. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, 2020. 5

[15] Federal Trade Commission. California company settles ftc allegations it deceived consumers about use of facial recognition in photo storage app, January 2021. 1

[16] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1322–1333, 2015. 1

[17] Regulation (EU) 2016/679 of the European parliament and of the council of 27 April 2016, 2016. 1

[18] Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *International Conference on Machine Learning*, pages 2242–2251, 2019. 5

[19] Antonio A Ginart, Melody Y Guan, Gregory Valiant, and James Zou. Making ai forget you: data deletion in machine learning. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 2019. 3

[20] Jairo Giraldo, Alvaro Cardenas, Murat Kantarcioglu, and Jonathan Katz. Adversarial classification under differential privacy. In *Network and Distributed Systems Security (NDSS) Symposium 2020*, 2020. 1

[21] Aditya Golatkar, Alessandro Achille, Avinash Ravichandran, Marzia Polito, and Stefano Soatto. Mixed-privacy forgetting in deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 792–801, 2021. 1, 4, 6

[22] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9304–9312, 2020. 1, 3, 6, 7, 12

[23] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Forgetting outside the box: Scrubbing deep networks of information accessible from input-output observations. In *European Conference on Computer Vision*, 2020. 1, 3, 6

[24] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations*, 2015. 1, 7

[25] Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, et al. Studying large language model generalization with influence functions. *arXiv preprint arXiv:2308.03296*, 2023. 9

[26] Chuan Guo, Tom Goldstein, Awni Y. Hannun, and Laurens van der Maaten. Certified data removal from machine learning models. In *Proceedings of the 37th International Conference on Machine Learning*, 2020. 1, 3, 4, 6, 7

[27] Jamie Hayes, Borja Balle, and M Pawan Kumar. Learning to be adversarially robust and differentially private. *arXiv preprint arXiv:2201.02265*, 2022. 1

[28] Dan Hendrycks, Kimin Lee, and Mantas Mazeika. Using pre-training can improve model robustness and uncertainty. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2712–2721, 09–15 Jun 2019. 6

[29] Dan Hendrycks, Kimin Lee, and Mantas Mazeika. Using pre-training can improve model robustness and uncertainty. In *International Conference on Machine Learning*, pages 2712–2721, 2019. 12

[30] Ruitong Huang, Bing Xu, Dale Schuurmans, and Csaba Szepesvári. Learning with a strong adversary. *CoRR*, abs/1511.03034, 2015. 1, 3

[31] Zachary Izzo, Mary Anne Smart, Kamalika Chaudhuri, and James Zou. Approximate data deletion from machine learning models. In *International Conference on Artificial Intelligence and Statistics*, pages 2008–2016, 2021. 1, 3

[32] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, 2017. 5, 7, 12

[33] Haowen Lin, Jian Lou, Li Xiong, and Cyrus Shahabi. Integer-arithmetic-only certified robustness for quantized neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7828–7837, 2021. 1

[34] Shen Lin, Xiaoyu Zhang, Chenyang Chen, Xiaofeng Chen, and Willy Susilo. Erm-ktp: Knowledge-level machine unlearning via knowledge transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. 3

[35] Junxu Liu, Jian Lou, Li Xiong, Jinfei Liu, and Xiaofeng Meng. Projected federated averaging with heterogeneous differential privacy. *Proceedings of the VLDB Endowment*, 15(4):828–840, 2021. 1

[36] Yang Liu, Mingyuan Fan, Cen Chen, Ximeng Liu, Zhuo Ma, Li Wang, and Jianfeng Ma. Backdoor defense with machine unlearning. In *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, page 280–289, 2022. 1

[37] Jian Lou and Yiu-ming Cheung. An uplink communication-efficient approach to featurewise distributed sparse optimization with differential privacy. *IEEE Transactions on Neural Networks and Learning Systems*, 32(10):4529–4543, 2020. 1

[38] Jie Ma, Xiangyuan Lan, Bineng Zhong, Guorong Li, Zhenjun Tang, Xianxian Li, and Rongrong Ji. Robust tracking via

uncertainty-aware semantic consistency. *IEEE Transactions on Circuits and Systems for Video Technology*, 33(4):1740–1751, 2022. 1

[39] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations*, 2018. 1, 7

[40] Ananth Mahadevan and Michael Mathioudakis. Certifiable machine unlearning for linear models. *arXiv preprint arXiv:2106.15093*, 2021. 3, 7

[41] Neil G Marchant, Benjamin IP Rubinstein, and Scott Alfeld. Hard to forget: Poisoning attacks on certified machine unlearning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7691–7700, 2022. 1

[42] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, 2017. 1

[43] Ronak Mehta, Sourav Pal, Vikas Singh, and Sathya N. Ravi. Deep unlearning via randomized conditionally independent hessians. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10422–10431, June 2022. 1, 3

[44] Seth Neel, Aaron Roth, and Saeed Sharifi-Malvajerdi. Descent-to-delete: Gradient-based methods for machine unlearning. In *Algorithmic Learning Theory*, 2021. 1

[45] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael P Wellman. Sok: Security and privacy in machine learning. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 399–414. IEEE, 2018. 1

[46] Jian Pei. A survey on data pricing: from economics to data science. *IEEE Transactions on knowledge and Data Engineering*, 2020. 5

[47] Hai Phan, My T Thai, Han Hu, Ruoming Jin, Tong Sun, and Dejing Dou. Scalable differential privacy with certified robustness in adversarial learning. In *International Conference on Machine Learning*, pages 7683–7694, 2020. 1

[48] NhatHai Phan, Minh N. Vu, Yang Liu, Ruoming Jin, Dejing Dou, Xintao Wu, and My T. Thai. Heterogeneous gaussian mechanism: Preserving differential privacy in deep learning with provable robustness. In Sarit Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 4753–4759, 2019. 1

[49] Ayush Sekhari, Jayadev Acharya, Gautam Kamath, and Ananda Theertha Suresh. Remember what you want to forget: Algorithms for machine unlearning. *Advances in Neural Information Processing Systems*, 34:18075–18086, 2021. 3, 4

[50] Uri Shaham, Yutaro Yamada, and Sahand Negahban. Understanding adversarial training: Increasing local stability of supervised models through robust optimization. *Neurocomputing*, 307:195–204, 2018. 3

[51] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy*, pages 3–18. IEEE, 2017. 1

[52] Liwei Song, Reza Shokri, and Prateek Mittal. Privacy risks of securing machine learning models against adversarial examples. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019. 1

[53] Qiheng Sun, Xiang Li, Jiayao Zhang, Li Xiong, Weiran Liu, Jinfei Liu, Zhan Qin, and Kui Ren. Shapleyfl: Robust federated learning based on shapley value. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2096–2108, 2023. 1

[54] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations*, 2014. 1

[55] Farnaz Tahmasebian, Jian Lou, and Li Xiong. Robustfed: a truth inference approach for robust federated learning. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2022. 1

[56] Pengfei Tang, Wenjie Wang, Jian Lou, and Li Xiong. Generating adversarial examples with distance constrained adversarial imitation networks. *IEEE Transactions on Dependable and Secure Computing*, 19(6):4145–4155, 2021.

[57] Anvith Thudi, Gabriel Deza, Varun Chandrasekaran, and Nicolas Papernot. Unrolling sgd: Understanding factors influencing machine unlearning. In *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*, pages 303–319. IEEE, 2022. 4

[58] Anvith Thudi, Hengrui Jia, Ilia Shumailov, and Nicolas Papernot. On the necessity of auditable algorithmic definitions for machine unlearning. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 4007–4022, 2022. 1

[59] Boxi Wu, Jindong Gu, Zhifeng Li, Deng Cai, Xiaofei He, and Wei Liu. Towards efficient adversarial training on vision transformers. In *European Conference on Computer Vision*, pages 307–325, 2022. 1

[60] Yinjun Wu, Edgar Dobriban, and Susan Davidson. Deltagrad: Rapid retraining of machine learning models. In *International Conference on Machine Learning*, 2020. 1, 4

[61] Jingwen Ye, Yifang Fu, Jie Song, Xingyi Yang, Songhua Liu, Xin Jin, Mingli Song, and Xinchao Wang. Learning with recoverable forgetting. In *European Conference on Computer Vision*, pages 87–103, 2022. 1

[62] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. Adversarial examples: Attacks and defenses for deep learning. *IEEE transactions on neural networks and learning systems*, 30(9):2805–2824, 2019. 1

[63] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference*, 2016. 6

[64] Guojun Zhang, Kaiwen Wu, Pascal Poupart, and Yaoliang Yu. Newton-type methods for minimax optimization. *arXiv preprint arXiv:2006.14592*, 2020. 5

[65] Peng-Fei Zhang, Guangdong Bai, Zi Huang, and Xin-Shun Xu. Machine unlearning for image retrieval: A generative scrubbing approach. In *Proceedings of the 30th ACM International Conference on Multimedia*, 2022. 1

[66] Xiaoyu Zhang, Yulin Jin, Tao Wang, Jian Lou, and Xiaofeng Chen. Purifier: Plug-and-play backdoor mitigation for pre-trained models via anomaly activation suppression. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 4291–4299, 2022. 1

Appendix

In this Appendix, Appendix A provides experiment details and deferred experiment figures and tables; Appendix B provides the deferred proofs and the complete algorithm description. In addition, Appendix B also describes how to generalize *MUter* from successive *single datapoint* unlearning to successive *batch of datapoints* removal. Our source code can be found in Supplementary Material.

A. Experiment Details and Deferred Experiment Figures and Tables

A.1. Experiment Details

Our experiments are based on the Pytorch platform and run on RTX 3090. For the pretraining the Wide ResNet model on Downsampled ImageNet, we run on four GPU devices; For all other experiments, we run on a single GPU device.

We summarize the hyperparameters used for our adversarial training/finetuning in Table 3. For Neural Network Model, we conduct experiments on Wide ResNet 28-10 model using the datasets Lacuna-10 and CIFAR-10. We first perform adversarial pretraining on Lacuna-100 and Downsampled ImageNet datasets, both by SGD with momentum for the outer-loops. For the former, we fix the learning rate to 0.1, and then train for 80 epochs with momentum 0.9 and weight decay 0.0005. For the latter, we use the pretrained model in [29] as the pretrained model here, which has a similar training process. Then, we freeze all but the last layer to adversarially finetune the model on Lacuna-10 and CIFAR-10 datasets.

We summarize the hyperparameters required by *MUter* in Table 4. In the neural network model experiments, [32] points out that under the non-convex setting, the Hessian matrix on parameters $\boldsymbol{\omega}^*$ sometimes will not be positive definite. We follow [32] to add a damping term λ on the diagonal with $\lambda = 0.0001$.

Dataset	Model type	Learning rate	(Tuning) Epoch	FGSM	PGD		
				epsilon	epsilon	alpha	steps
MNIST-b	Logistic	0.01	100	0.25	0.25	0.03	15
	Ridge	0.01	100	0.25	0.25	0.03	15
Covtype	Logistic	0.1	15	4/255	4/255	0.004	7
	Ridge	0.1	15	4/255	4/255	0.004	7
Lacuna-10	Neural Network	0.01	20	8/255	8/255	2/255	10
CIFAR-10	Neural Network	0.001	10	8/255	8/255	2/255	10

Table 3: Adversarial training/finetuning parameters.

A.2. Datasets

Linear Model. We perform experiments on MNIST-b and Covtype. MNIST-b is taken from the MNIST dataset, which consists of $28 * 28$ grayscale images from digist '0' to digit '9'. We select the digit '1' and digit '7' to form the binary subset MNIST-b. Covtype with 54 attributes is used to classify the main tree species in the Roosevelt National Forest wilderness area, where use the binary classification version from LIBSVM.

Dataset	Model type	Neumann Series order k	Conjugate Gradient iterations C
MNIST-b	Logistic	3	10
	Ridge	3	10
Covtype	Logistic	3	20
	Ridge	20	20
Lacuna-10	Neural Network	100	10
CIFAR-10	Neural Network	100	20

Table 4: *MUter* parameters.

Neural Network Model. We introduce Lacuna-100 and downsampled ImageNet as core datasets, and conduct experiments on target datasets Lacuna-10 and CIFAR-10. CIFAR-10 consists of $32 * 32$ color pictures, covering different animals and machines in 10 categories. The Downsampled ImageNet is derived from the 1000-class ImageNet dataset, which resize the image to $32 * 32$. Lacuna-10/Lacuna-100 comes from [22]. We use the same data processing method to select 10/100 celebrities (no intersection) from VGGFace2 [7], and each celebrity randomly selects at least 500 pictures. Then each celebrity divides 100 pictures to form the test set, and the remaining images to form the training set. Finally, we resize the images to $32 * 32$.

To facilitate the constrained adversarial perturbation for adversarial training, all the above datasets are scaled to $[0, 1]$ (for image data, we use Totensor to transform, and for Covtype, we choose the version scaled to $[0, 1]$ in LIBSVM.). We summarize the dimensions, classes, and quantity information of the above datasets in Table 5.

Dataset	Domension	Classes	Train data	Test Data
MNIST-b	784	2	11,982	1,198
Covtype	54	2	522,910	58,102
Lacuna-10	3072	10	4,374	1,000
CIFAR-10	3072	10	50,000	10,000

Table 5: Datasets Statistics

A.3. Deferred Experiment Figures and Tables

In this section, we report additional experiment results, where Subsection A.3.1 reports additional experiment results under the PGD setting, and Subsection A.3.2 reports additional experiment results under the FGSM setting.

A.3.1 Deferred Experiment Figures under PGD Setting

We summarize the experiment results under Ridge Model with PGD in Figure 6.

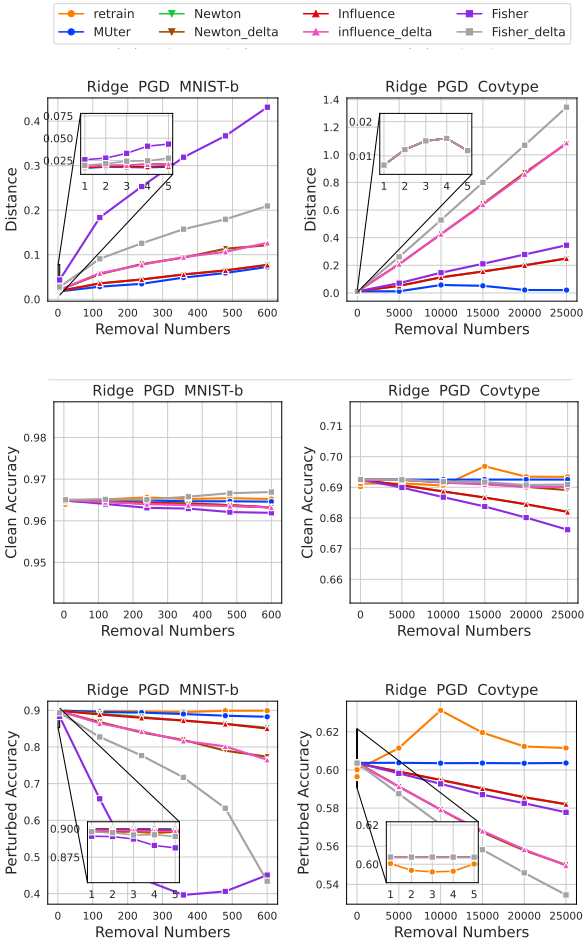

Figure 6: Evaluation results on Ridge Regression Model with PGD: Effectiveness (left column), Accuracy (middle column), and Robustness (right column) on datasets MNIST (top) and Covtype (bottom). Large plots have greater removal numbers: $1\%, 2\%, \cdots, 5\%$; Small plots inside the large plots have fewer removal numbers: $1, 2, \cdots, 5$.

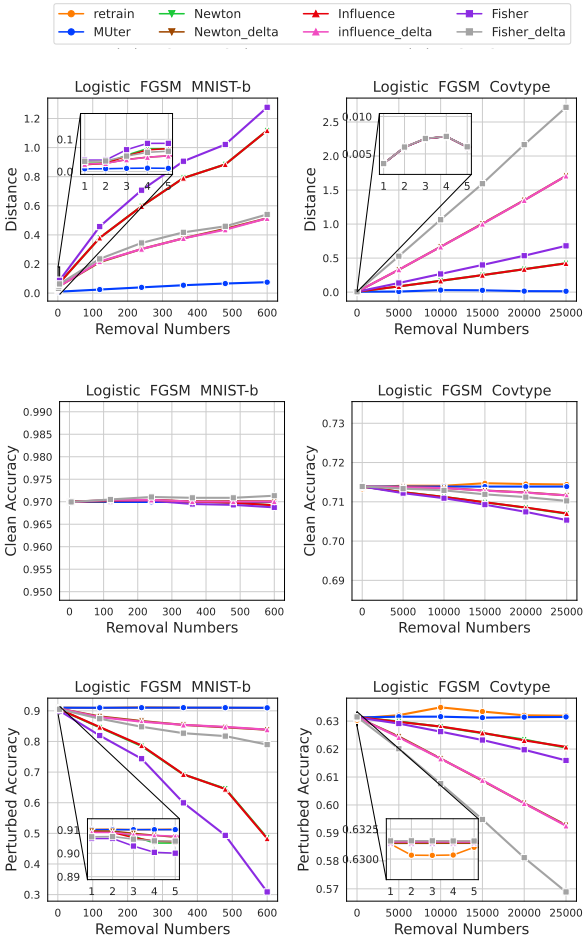

Figure 7: Evaluation results on Logistic Regression Model with FGSM: Effectiveness (left column), Accuracy (middle column), and Robustness (right column) on datasets MNIST (top) and Covtype (bottom). Large plots have greater removal numbers: $1\%, 2\%, \cdots, 5\%$; Small plots inside the large plots have fewer removal numbers: $1, 2, \cdots, 5$.

A.3.2 Deferred Experiment Figures and Tables under FGSM Setting

In this part, we change the way of generating perturbations from PGD to FGSM. With the same experimental settings under PGD conditions, we report experiment results both on Linear Model and Neural Network Model.

Results with Linear Model. We report the effectiveness, accuracy and robustness metrics of logistic model and ridge model in Figure 7 and Figure 8, respectively. In Table 6, we summarize the efficiency comparison for linear model under FGSM setting.

Model Type	Removal Number	MNIST-b Fisher	MNIST-b F-delta	MNIST-b *MUter*	MNIST-b Retrain	Covtype Fisher	Covtype F-delta	Covtype *MUter*	Covtype Retrain
LR	1	0.002	0.002	0.008	14.6	0.002	0.002	0.007	75
	5	0.010	0.012	0.045	14.5	0.008	0.009	0.033	76
	10	0.020	0.022	0.089	14.2	0.018	0.019	0.065	77
	1%	0.242	0.249	0.978	14.3	9.178	11.278	33.974	77
RR	1	0.002	0.002	0.008	14.6	0.002	0.002	0.007	78
	5	0.009	0.012	0.047	14.5	0.009	0.009	0.034	78
	10	0.018	0.020	0.091	14.3	0.018	0.021	0.063	79
	1%	0.239	0.244	0.993	14.5	9.797	11.043	33.505	79

Table 6: Efficiency results with Logistic Regression Model (top) and Ridge Regression Model (bottom) under FGSM: The unlearning time (in seconds) of **Fisher**, **Fisher-delta**, *MUter* and **Retrain** under varying removal numbers: 1, 5, 10, 1% (120 for MNSIT-b, 5000 for Covtype).

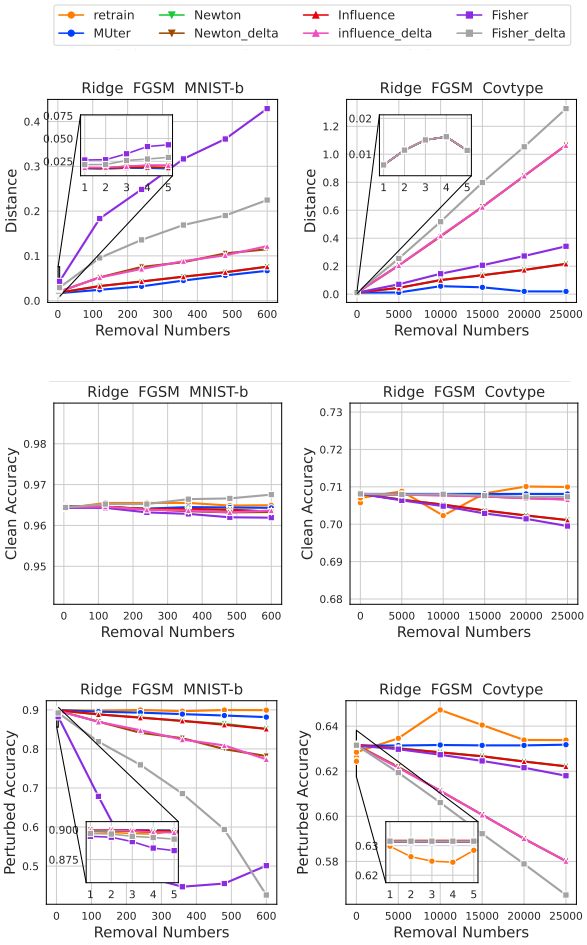

Figure 8: Evaluation results on Ridge Regression Model with FGSM: Effectiveness (left column), Accuracy (middle column), and Robustness (right column) on datasets MNIST (top) and Covtype (bottom). Large plots have greater removal numbers: $1\%, 2\%, \cdots, 5\%$; Small plots inside the large plots have fewer removal numbers: $1, 2, \cdots, 5$.

Result with Neural Network Model. We report the effectiveness, accuracy, and robustness metrics of the neural network model in Figure 9. In Table 7, we summarize the efficiency comparison for the neural network model under the FGSM setting.

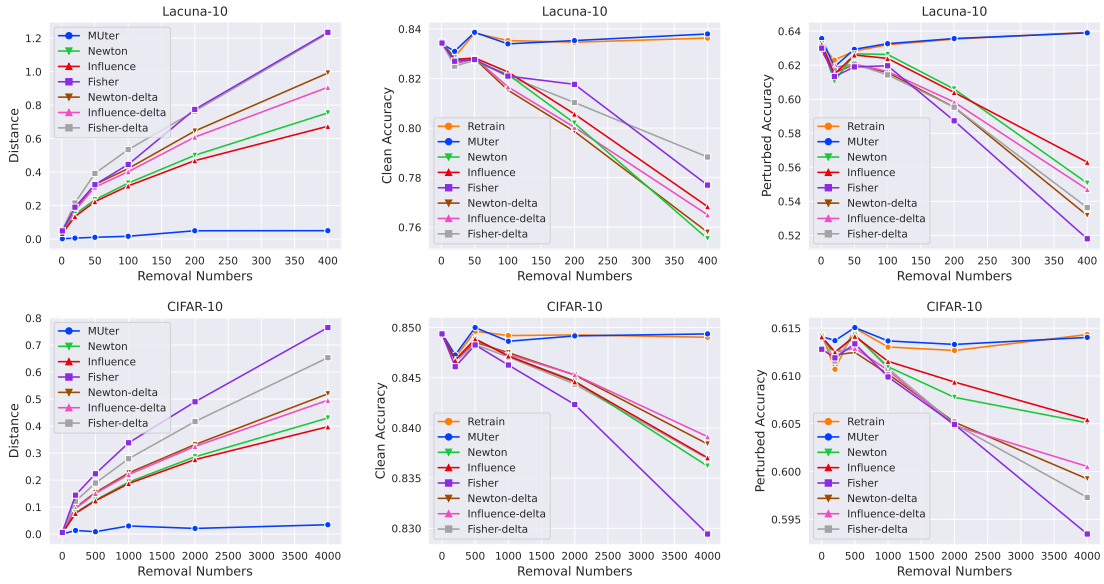

Figure 9: Evaluation results on Neural Network with FGSM: Effectiveness (left column), Accuracy (middle column), and Robustness (right column) on datasets Lacuna-10 (top) and CIFAR-10 (bottom), under removal numbers $(1, 0.4\%, 1\%, 2\%, 4\%, 8\%)$.

Removal	**Lacuna-10**				**CIFAR-10**			
Number	Fisher	F-delta	*MUter*	Retrain	Fisher	F-delta	*MUter*	Retrain
1	1.51	1.57	3.96	150	1.43	1.56	3.72	799
5	7.81	8.08	20.72	151	7.71	8.11	18.23	794
10	15.46	15.94	39.82	150	15.42	15.87	36.84	790

Table 7: Efficiency results with Neural Network Model under FGSM: The unlearning time (in seconds) of **Fisher**, **Fisher-delta**, *MUter* and **Retrain** under varying removal numbers: 1, 5, 10.

B. Deferred Proofs and Algorithm Description

In this section, Subsection B.1 provides the omitted proofs for the derivation of the ATM unlearning update (i.e., Lemma 1 and Theorem 1), Subsection B.2 provides the omitted proof for the derivation of the successive unlearning setting (i.e., Corollary 1), Subsection B.3 provides the deferred Lemma for Schur complement for completeness and the omitted proof for Theorem 2, Subsection B.4 provides the complete algorithm description for *MUter* , and Subsection B.5 how to generalize *MUter* from successive *single datapoint* unlearning to successive *batch of datapoints* removal.

B.1. Proofs for Derivation of ATM Unlearning Update

B.1.1 Proof of Lemma 1

Proof. By Taylor expansion around both $\boldsymbol{\omega}^*$ and $\delta_i(\boldsymbol{\omega}^*)$, we have

$$\begin{aligned}
&\frac{1}{n-1}\sum_{i=1,i\neq i^\dagger}^{n}\nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^u, \mathbf{x}_i+\delta_i(\boldsymbol{\omega}^u))\\
=&\frac{1}{n-1}\sum_{i=1,i\neq i^\dagger}^{n}\Big[\nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i+\delta_i(\boldsymbol{\omega}^*))\\
&+\partial_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i+\delta_i(\boldsymbol{\omega}^*))(\boldsymbol{\omega}^u-\boldsymbol{\omega}^*)+O(\|\boldsymbol{\omega}^u-\boldsymbol{\omega}^*\|_2^2)\\
&+\partial_{\boldsymbol{\omega}\boldsymbol{\delta}} l(\boldsymbol{\omega}^*, \mathbf{x}_i+\delta_i(\boldsymbol{\omega}^*))(\delta_i(\boldsymbol{\omega}^u)-\delta_i(\boldsymbol{\omega}^*))+O(\|\delta_i(\boldsymbol{\omega}^u)-\delta_i(\boldsymbol{\omega}^*)\|_2^2)\Big].
\end{aligned}\tag{17}$$

By furthering neglecting the higher-order Taylor expansion terms $O(\|\boldsymbol{\omega}^u-\boldsymbol{\omega}^*\|_2^2)$ and $O(\|\delta_i(\boldsymbol{\omega}^u)-\delta_i(\boldsymbol{\omega}^*)\|_2^2)$, we have the following relationship,

$$\begin{aligned}
&\frac{1}{n-1}\sum_{i=1,i\neq i^\dagger}^{n}\nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^u, \mathbf{x}_i+\delta_i(\boldsymbol{\omega}^u))\\
\approx&\frac{1}{n-1}\sum_{i=1,i\neq i^\dagger}^{n}\Big[\nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i+\delta_i(\boldsymbol{\omega}^*))\\
&+\partial_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i+\delta_i(\boldsymbol{\omega}^*))(\boldsymbol{\omega}^u-\boldsymbol{\omega}^*)\\
&+\partial_{\boldsymbol{\omega}\boldsymbol{\delta}} l(\boldsymbol{\omega}^*, \mathbf{x}_i+\delta_i(\boldsymbol{\omega}^*))(\delta_i(\boldsymbol{\omega}^u)-\delta_i(\boldsymbol{\omega}^*))\Big],
\end{aligned}\tag{18}$$

which proves Lemma 1. □

B.1.2 Proof of Theorem 1

Proof. By Lemma 1, we begin with

$$\begin{aligned}
&\frac{1}{n-1}\sum_{i=1,i\neq i^\dagger}^{n}\nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^u, \mathbf{x}_i+\delta_i(\boldsymbol{\omega}^u))\\
\approx&\frac{1}{n-1}\sum_{i=1,i\neq i^\dagger}^{n}\Big[\nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i+\delta_i(\boldsymbol{\omega}^*))\\
&+\partial_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i+\delta_i(\boldsymbol{\omega}^*))(\boldsymbol{\omega}^u-\boldsymbol{\omega}^*)+\partial_{\boldsymbol{\omega}\boldsymbol{\delta}} l(\boldsymbol{\omega}^*, \mathbf{x}_i+\delta_i(\boldsymbol{\omega}^*))(\delta_i(\boldsymbol{\omega}^u)-\delta_i(\boldsymbol{\omega}^*))\Big].
\end{aligned}\tag{19}$$

For the first line on the right hand side of eq.(19), we have

$$\begin{aligned}
&\frac{1}{n-1}\sum_{i=1,i\neq i^\dagger}^{n}\nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i+\delta_i(\boldsymbol{\omega}^*))\\
=&\frac{1}{n-1}\sum_{i=1}^{n}\nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i+\delta_i(\boldsymbol{\omega}^*)) - \frac{1}{n-1}\nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_{i^\dagger}+\delta_{i^\dagger}(\boldsymbol{\omega}^*))\\
=&0-\frac{1}{n-1}\nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_{i^\dagger}+\delta_{i^\dagger}(\boldsymbol{\omega}^*)\\
=&-\frac{1}{n-1}\nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_{i^\dagger}+\delta_{i^\dagger}(\boldsymbol{\omega}^*)),
\end{aligned} \tag{20}$$

where the second equality is obtained as follows. Since $\boldsymbol{\omega}^*$ is the optimum of the following ATM,

$$\boldsymbol{\omega}^* = \operatorname*{argmin}_{\boldsymbol{\omega}} \max_{\boldsymbol{\delta}_i\in\mathcal{B}(\mathbf{x}_i,r)} \frac{1}{n}\sum_{i=1}^{n} l(\boldsymbol{\omega}, \mathbf{x}_i+\delta(\boldsymbol{\omega}^*)), \tag{21}$$

it satisfies the following by Danskin's Theorem,

$$\frac{1}{n}\sum_{i=1}^{n}\nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i+\delta_i(\boldsymbol{\omega}^*)) = 0, \tag{22}$$

which is the second equality relation in eq.(20).

For the second line on the right hand side of eq.(19), we further expand $(\delta_i(\boldsymbol{\omega}^u)-\delta_i(\boldsymbol{\omega}^*))$ by the implicit function theorem. That is, for all $i\in\{1,\cdots,n\}$, we have

$$\delta_i(\boldsymbol{\omega}^u)-\delta_i(\boldsymbol{\omega}^*) = -\partial^{-1}_{\boldsymbol{\delta}\boldsymbol{\delta}} l(\boldsymbol{\omega}^*, \mathbf{x}_i+\delta_i(\boldsymbol{\omega}^*))\cdot\partial_{\boldsymbol{\delta}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i+\delta_i(\boldsymbol{\omega}^*))\cdot(\boldsymbol{\omega}^u-\boldsymbol{\omega}^*), \tag{23}$$

which gives

$$\begin{aligned}
&\frac{1}{n-1}\sum_{i=1,i\neq i^\dagger}^{n}\Big[\partial_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i+\delta_i(\boldsymbol{\omega}^*))(\boldsymbol{\omega}^u-\boldsymbol{\omega}^*) + \partial_{\boldsymbol{\omega}\boldsymbol{\delta}} l(\boldsymbol{\omega}^*, \mathbf{x}_i+\delta_i(\boldsymbol{\omega}^*))(\delta_i(\boldsymbol{\omega}^u)-\delta_i(\boldsymbol{\omega}^*))\Big]\\
=&\frac{1}{n-1}\sum_{i=1,i\neq i^\dagger}^{n}\Big[\partial_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i+\delta_i(\boldsymbol{\omega}^*))(\boldsymbol{\omega}^u-\boldsymbol{\omega}^*)\\
&-\partial_{\boldsymbol{\omega}\boldsymbol{\delta}} l(\boldsymbol{\omega}^*, \mathbf{x}_i+\delta_i(\boldsymbol{\omega}^*))\partial^{-1}_{\boldsymbol{\delta}\boldsymbol{\delta}} l(\boldsymbol{\omega}^*, \mathbf{x}_i+\delta_i(\boldsymbol{\omega}^*))\partial_{\boldsymbol{\delta}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i+\delta_i(\boldsymbol{\omega}^*))(\boldsymbol{\omega}^u-\boldsymbol{\omega}^*)\Big]\\
=&\frac{1}{n-1}\sum_{i=1,i\neq i^\dagger}^{n}\mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i+\delta_i(\boldsymbol{\omega}^*))(\boldsymbol{\omega}^u-\boldsymbol{\omega}^*).
\end{aligned} \tag{24}$$

By substituting eq.(20) and eq.(24) into eq.(19), we have

$$\begin{aligned}
&\frac{1}{n-1}\sum_{i=1,i\neq i^\dagger}^{n}\nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^u, \mathbf{x}_i+\delta_i(\boldsymbol{\omega}^u))\\
\approx&-\frac{1}{n-1}\nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_{i^\dagger}+\delta_{i^\dagger}(\boldsymbol{\omega}^*)) + \frac{1}{n-1}\sum_{i=1,i\neq i^\dagger}^{n}\mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i+\delta_i(\boldsymbol{\omega}^*))(\boldsymbol{\omega}^u-\boldsymbol{\omega}^*).
\end{aligned} \tag{25}$$

Thus, $\boldsymbol{\omega}^u = \boldsymbol{\omega}^* + \mathcal{U}(\boldsymbol{\omega}^*, i^\dagger) = \boldsymbol{\omega}^* + \Big[\sum_{i=1,i\neq i^\dagger}^{n}\mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i+\delta_i(\boldsymbol{\omega}^*))\Big]^{-1}\cdot\Big[\nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_{i^\dagger}+\delta_{i^\dagger}(\boldsymbol{\omega}^*))\Big]$ is the the solution to the following linear system,

$$-\frac{1}{n-1}\nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_{i^\dagger}+\delta_{i^\dagger}(\boldsymbol{\omega}^*)) + \frac{1}{n-1}\sum_{i=1,i\neq i^\dagger}^{n}\mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i+\delta_i(\boldsymbol{\omega}^*))(\boldsymbol{\omega}^u-\boldsymbol{\omega}^*) = 0, \tag{26}$$

which gives $\frac{1}{n-1}\sum_{i=1,i\neq i^\dagger}^{n}\nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^u, \mathbf{x}_i+\delta_i(\boldsymbol{\omega}^u)) \approx 0$ according to eq.(25). As a result, we have proved that $\boldsymbol{\omega}^u$ satisfies the approximate unlearning criteria for ATM. □

B.2. Proofs for derivation of Successive Unlearning Setting

B.2.1 Proof of Corollary 1

Proof. Denote the data that have been deleted by $\mathcal{U}_r := \{i_1^\dagger, i_2^\dagger, \ldots, i_r^\dagger\}$. Similar to the ATM unlearning standard in eq.(3), we have the retraining-from-scratch model parameter after the $r+1$-th unlearning by $\boldsymbol{\omega}^*_{-\mathcal{U}_r \cup i^\dagger}$, which has the following definition,

$$\boldsymbol{\omega}^*_{-\mathcal{U}_r \cup i^\dagger} = \underset{\boldsymbol{\omega}}{\operatorname{argmin}} \frac{1}{n-r-1} \sum_{i=1, i \neq \mathcal{U}_r \cup i^\dagger}^{n} \max_{\boldsymbol{\delta}_i \in \mathcal{B}(\mathbf{x}_i, r)} l(\boldsymbol{\omega}, \mathbf{x}_i + \boldsymbol{\delta}_i). \tag{27}$$

The unlearning criteria for ATM at the $r+1$-th unlearning is as follows,

$$\sum_{i=1, i \neq \mathcal{U}_r \cup i^\dagger}^{n} \nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*_{-\mathcal{U}_r \cup i^\dagger}, \mathbf{x}_i + \delta_i(\boldsymbol{\omega}^*_{-\mathcal{U}_r \cup i^\dagger})) \approx 0. \tag{28}$$

Our aim is to show that $\boldsymbol{\omega}^u_{r+1}$ approximately satisfies the above criteria,

$$\sum_{i=1, i \neq \mathcal{U}_r \cup i^\dagger}^{n} \nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^u_{r+1}, \mathbf{x}_i + \delta_i(\boldsymbol{\omega}^u_{r+1})) \approx 0, \tag{29}$$

which is also the approximate unlearning criteria for ATM in eq.(6).

The proof is similar to Theorem 1, as follows,

$$\begin{aligned}
&\sum_{i=1, i \neq \mathcal{U}_r \cup i^\dagger}^{n} \nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^u_{r+1}, \mathbf{x}_i + \delta_i(\boldsymbol{\omega}^u_{r+1})) \\
=&\Big[\sum_{i=1}^{n} \nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^u_{r+1}, \mathbf{x}_i + \delta_i(\boldsymbol{\omega}^u_{r+1}))\Big] - \Big[\sum_{i=i_1^\dagger}^{i_r^\dagger} \nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^u_{r+1}, \mathbf{x}_i + \delta_i(\boldsymbol{\omega}^u_{r+1}))\Big] - \Big[\nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^u_{r+1}, \mathbf{x}_{i^\dagger} + \delta_{i^\dagger}(\boldsymbol{\omega}^u_{r+1}))\Big] \\
\approx&\Big[\Big(\sum_{i=1}^{n} \nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i + \delta_i(\boldsymbol{\omega}^*)) + \partial_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i + \delta_i(\boldsymbol{\omega}^*))(\boldsymbol{\omega}^u_{r+1} - \boldsymbol{\omega}^*) + \partial_{\boldsymbol{\omega}\boldsymbol{\delta}} l(\boldsymbol{\omega}^*, \mathbf{x}_i + \delta_i(\boldsymbol{\omega}^*))(\delta_i(\boldsymbol{\omega}^u_{r+1}) - \delta_i(\boldsymbol{\omega}^*))\Big)\Big] \\
&- \Big[\Big(\sum_{i=i_1^\dagger}^{i_r^\dagger} \nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i + \delta_i(\boldsymbol{\omega}^*)) + \partial_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i + \delta_i(\boldsymbol{\omega}^*))(\boldsymbol{\omega}^u_{r+1} - \boldsymbol{\omega}^*) + \partial_{\boldsymbol{\omega}\boldsymbol{\delta}} l(\boldsymbol{\omega}^*, \mathbf{x}_i + \delta_i(\boldsymbol{\omega}^*))(\delta_i(\boldsymbol{\omega}^u_{r+1}) - \delta_i(\boldsymbol{\omega}^*))\Big)\Big] \\
&- \Big[\Big(\nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_{i^\dagger} + \delta_{i^\dagger}(\boldsymbol{\omega}^*)) + \partial_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_{i^\dagger} + \delta_{i^\dagger}(\boldsymbol{\omega}^*))(\boldsymbol{\omega}^u - \boldsymbol{\omega}^*) + \partial_{\boldsymbol{\omega}\boldsymbol{\delta}} l(\boldsymbol{\omega}^*, \mathbf{x}_{i^\dagger} + \delta_{i^\dagger}(\boldsymbol{\omega}^*))(\delta_{i^\dagger}(\boldsymbol{\omega}^u_{r+1}) - \delta_{i^\dagger}(\boldsymbol{\omega}^*))\Big)\Big] \\
=&\Big[\sum_{i=1}^{n} \nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i + \delta_i(\boldsymbol{\omega}^*)) + \mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i + \delta_i(\boldsymbol{\omega}^*))(\boldsymbol{\omega}^u_{r+1} - \boldsymbol{\omega}^*)\Big] \\
&- \Big[\sum_{i=i_1^\dagger}^{i_r^\dagger} \nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i + \delta_i(\boldsymbol{\omega}^*)) + \mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i + \delta_i(\boldsymbol{\omega}^*))(\boldsymbol{\omega}^u_{r+1} - \boldsymbol{\omega}^*)\Big] \\
&- \Big[\nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_{i^\dagger} + \delta_{i^\dagger}(\boldsymbol{\omega}^*)) + \mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_{i^\dagger} + \delta_{i^\dagger}(\boldsymbol{\omega}^*))(\boldsymbol{\omega}^u_{r+1} - \boldsymbol{\omega}^*)\Big] \\
=&\Big[\sum_{i=1}^{n} \mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i + \delta_i(\boldsymbol{\omega}^*))(\boldsymbol{\omega}^u_{r+1} - \boldsymbol{\omega}^*)\Big] \\
&- \Big[\sum_{i=i_1^\dagger}^{i_r^\dagger} \nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i + \delta_i(\boldsymbol{\omega}^*)) + \mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i + \delta_i(\boldsymbol{\omega}^*))(\boldsymbol{\omega}^u_{r+1} - \boldsymbol{\omega}^*)\Big] \\
&- \Big[\nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_{i^\dagger} + \delta_{i^\dagger}(\boldsymbol{\omega}^*)) + \mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_{i^\dagger} + \delta_{i^\dagger}(\boldsymbol{\omega}^*))(\boldsymbol{\omega}^u_{r+1} - \boldsymbol{\omega}^*)\Big],
\end{aligned} \tag{30}$$

where the approximation equality is by the Taylor expansion and neglecting the higher-order terms, the second equality is by implicit function theorem and the definition of the total Hessian, and the third equality is by $\sum_{i=1}^{n} \nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i + \delta_i(\boldsymbol{\omega}^*)) = 0$ by eq.(22).

Thus, $\boldsymbol{\omega}_{r+1}^u = \boldsymbol{\omega}^* + \boldsymbol{\mathcal{U}}_r(\boldsymbol{\omega}^*, i^\dagger)$ is the the solution to the following linear system,

$$
\begin{aligned}
&- \sum_{i=i_1^\dagger}^{i_r^\dagger} \nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i + \delta_i(\boldsymbol{\omega}^*)) - \nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_{i^\dagger} + \delta_{i^\dagger}(\boldsymbol{\omega}^*)) \\
&+ \Big[\sum_{i=1}^{n} \mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i + \delta_i(\boldsymbol{\omega}^*)) - \sum_{i=i_1^\dagger}^{i_r^\dagger} \mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_i + \delta_i(\boldsymbol{\omega}^*)) - \mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*, \mathbf{x}_{i^\dagger} + \delta_{i^\dagger}(\boldsymbol{\omega}^*)) \Big] (\boldsymbol{\omega}^u - \boldsymbol{\omega}^*) = 0,
\end{aligned}
\tag{31}
$$

which gives $\sum_{i=1, i \neq \mathcal{U}_r \cup i^\dagger}^{n} \nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}_{r+1}^u, \mathbf{x}_i + \delta_i(\boldsymbol{\omega}_{r+1}^u)) \approx 0$. As a result, we have proved that $\boldsymbol{\omega}^u$ satisfies the approximate unlearning criteria for ATM in eq.(29) that is consistent with eq.(6). □

B.3. Proof for Schur Complement Conversion

B.3.1 Additional Lemma of Schur Complement

Lemma 2. *(Schur Complement Conversion) Let* $\mathbf{S} = \mathbf{H}_{11} - \mathbf{H}_{12}\mathbf{H}_{22}^{-1}\mathbf{H}_{21}$. *If* $\mathbf{H}_{22}$ *and* $\mathbf{S}$ *are invertible, then* $\mathbf{H}$ *is invertible and the following relation holds,*

$$
\mathbf{H}^{-1} = \begin{bmatrix} \mathbf{H}_{11} & \mathbf{H}_{12} \\ \mathbf{H}_{21} & \mathbf{H}_{22} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{S}^{-1} & -\mathbf{S}^{-1}\mathbf{H}_{12}\mathbf{H}_{22}^{-1} \\ -\mathbf{H}_{22}\mathbf{H}_{21}\mathbf{S}^{-1} & \mathbf{H}_{22}^{-1}\mathbf{H}_{21}\mathbf{S}^{-1}\mathbf{H}_{12}\mathbf{H}_{22}^{-1} \end{bmatrix}. \tag{32}
$$

B.3.2 Proof of Theorem 2

Proof. Let

$$
\mathbf{S} = \Big[\mathcal{M}_r[\mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}}] - \partial_{\boldsymbol{\omega}\boldsymbol{\omega}} l_{i^\dagger}^* \Big] - \Big[-\partial_{\boldsymbol{\omega}\boldsymbol{\delta}} \partial_{\boldsymbol{\delta}\boldsymbol{\delta}}^{-1} \partial_{\boldsymbol{\delta}\boldsymbol{\omega}} l_{i^\dagger}^* \Big], \tag{33}
$$

which is the Schur complement of the following block matrix,

$$
\mathbf{H} = \begin{bmatrix} \mathbf{H}_{11} & \mathbf{H}_{12} \\ \mathbf{H}_{21} & \mathbf{H}_{22} \end{bmatrix} = \begin{bmatrix} \mathcal{M}_r[\mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}}] - \partial_{\boldsymbol{\omega}\boldsymbol{\omega}} l_{i^\dagger}^* & \partial_{\boldsymbol{\omega}\boldsymbol{\delta}} l_{i^\dagger}^* \\ \partial_{\boldsymbol{\delta}\boldsymbol{\omega}} l_{i^\dagger}^* & -\partial_{\boldsymbol{\delta}\boldsymbol{\delta}} l_{i^\dagger}^* \end{bmatrix}. \tag{34}
$$

Then, based on Lemma 2, we have

$$
\mathbf{S}^{-1}\mathbf{g} = \Delta\boldsymbol{\omega} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{H}_{11} & \mathbf{H}_{12} \\ \mathbf{H}_{21} & \mathbf{H}_{22} \end{bmatrix}^{-1} \cdot \begin{bmatrix} \mathbf{g} \\ \mathbf{0} \end{bmatrix}, \tag{35}
$$

which can be cast as the solution of the following linear system,

$$
\begin{bmatrix} \mathbf{H}_{11} & \mathbf{H}_{12} \\ \mathbf{H}_{21} & \mathbf{H}_{22} \end{bmatrix} \cdot \begin{bmatrix} \Delta\boldsymbol{\omega} \\ \Delta\boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} \mathbf{g} \\ \mathbf{0} \end{bmatrix}. \tag{36}
$$

By substituting eq.(34) in, we have

$$
\begin{bmatrix} \mathcal{M}_r[\mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}}] - \partial_{\boldsymbol{\omega}\boldsymbol{\omega}} l_{i^\dagger}^* & \partial_{\boldsymbol{\omega}\boldsymbol{\delta}} l_{i^\dagger}^* \\ \partial_{\boldsymbol{\delta}\boldsymbol{\omega}} l_{i^\dagger}^* & -\partial_{\boldsymbol{\delta}\boldsymbol{\delta}} l_{i^\dagger}^* \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{\omega} \\ \Delta\boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} \mathcal{M}_r[\nabla_{\boldsymbol{\omega}}] + \nabla_{\boldsymbol{\omega}} l_{i^\dagger}^* \\ \mathbf{0} \end{bmatrix}, \tag{37}
$$

which proves Theorem 2. □

B.4. The Complete Algorithm Description

We present the complete algorithm description for *MUter* in Algorithm 1.

Algorithm 1 The Complete Algorithm Description for *MUter*

Input: Training dataset $\mathcal{D}=\{(\mathbf{x}_1,y_1),\dots,(\mathbf{x}_n,y_n)\}$, loss l, adversarial perturbation constraint $\mathcal{B}(\mathbf{x}_i,r)$, adversarial training finishing status $\{\boldsymbol{\omega}^*,\delta_1(\boldsymbol{\omega}^*),\cdots,\delta_n(\boldsymbol{\omega}^*)\}$, a sequence of indices to be removed $\mathcal{U}=\{i_1^\dagger,i_2^\dagger,\dots,\}$

Stage I. Pre-Unlearning:

Compute memory $\mathcal{M}_0[\mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}}]=\sum_{i=1}^n \widetilde{\mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}}} l(\boldsymbol{\omega}^*,\mathbf{x}_i+\delta_i(\boldsymbol{\omega}^*))$ by eq.(16);

Initialize $\mathcal{M}_0[\nabla_{\boldsymbol{\omega}}]=\mathbf{0}$ and $\mathcal{M}_0[\boldsymbol{\omega}^*]=\boldsymbol{\omega}^*$;

for $i_{r+1}^\dagger=i^\dagger$ **do**

  Stage II. Unlearning:

  Re-compute the adversarial perturbation $\delta_{i^\dagger}(\boldsymbol{\omega}^*)$;

  Compute gradient $\mathbf{g}=\nabla_{\boldsymbol{\omega}} l_{i^\dagger}^*$;

  Apply conjugate gradient (with C iterations) to solve the least square problem of the linear system

$$\begin{bmatrix}\mathcal{M}_r[\mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}}]-\partial_{\boldsymbol{\omega}\boldsymbol{\omega}} l_{i^\dagger}^* & \partial_{\boldsymbol{\omega}\boldsymbol{\delta}} l_{i^\dagger}^* \\ \partial_{\boldsymbol{\delta}\boldsymbol{\omega}} l_{i^\dagger}^* & -\partial_{\boldsymbol{\delta}\boldsymbol{\delta}} l_{i^\dagger}^*\end{bmatrix}\begin{bmatrix}\Delta\boldsymbol{\omega}\\ \Delta\boldsymbol{\alpha}\end{bmatrix}=\begin{bmatrix}\mathcal{M}_r[\nabla_{\boldsymbol{\omega}}]+\nabla_{\boldsymbol{\omega}} l_{i^\dagger}^*\\ \mathbf{0}\end{bmatrix},$$

  Obtain $\mathfrak{U}_r(\boldsymbol{\omega}^*,i_{r+1}^\dagger)=\Delta\boldsymbol{\omega}$, the model paraeter after the $r+1$-th unlearning: $\boldsymbol{\omega}_{r+1}^u=\boldsymbol{\omega}^*+\mathfrak{U}_r(\boldsymbol{\omega}^*,i_{r+1}^\dagger)$;

  Stage III. Post-Unlearning:

  $\mathbf{m}_{i^\dagger}=\partial_{\boldsymbol{\omega}\boldsymbol{\omega}} l_{i^\dagger}^*-\partial_{\boldsymbol{\omega}\boldsymbol{\delta}} l_{i^\dagger}^*\big[\sum_{j=0}^k(\mathbf{I}-\partial_{\boldsymbol{\delta}\boldsymbol{\delta}} l_{i^\dagger}^*)^j\big]\partial_{\boldsymbol{\delta}\boldsymbol{\omega}} l_{i^\dagger}^*$ by eq.(16);

  Update the memory $\mathcal{M}_{r+1}[\mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}}]=\mathcal{M}_r[\mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}}]-\mathbf{m}_{i^\dagger}$ and $\mathcal{M}_{r+1}[\nabla_{\boldsymbol{\omega}}]=\mathcal{M}_r[\nabla_{\boldsymbol{\omega}}]+\mathbf{g}$;

end for

B.5. Extension to Successive Batch Unlearning

In this subsection, we show that our method can be generalized to the successive batch unlearning setting.

Successive Batch Unlearning Setting. Denote the index sets of datapoints that have already been forgotten at timestamp r by $\mathcal{U}_1^\dagger,\cdots,\mathcal{U}_r^\dagger\}$, where each $\mathcal{U}_r^\dagger$ contains a set of data indices. Let the set of datapoints to be forgotten at timestamp $r+1$ by $\mathcal{U}_{r+1}^\dagger=\mathcal{U}^\dagger$. Corollary 2 below extends Corollary 1 to support successive batch unlearning for ATM.

Corollary 2. *Considering the successive batch unlearning setting, let the machine unlearning update at the $r+1$-th timestamp $\mathfrak{U}_r(\boldsymbol{\omega}^*,\mathcal{U}^\dagger)$ take the following form*

$$\begin{aligned}\mathfrak{U}_r(\boldsymbol{\omega}^*,i_{r+1}^\dagger):=\Bigg\{&\overbrace{\Big[\sum_{i=1}^n \mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*,\mathbf{x}_i+\boldsymbol{\delta}_i^*)\Big]}^{\mathcal{M}_0[\mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}}]}-\\ &\overbrace{\Big[\sum_{j=1}^r\sum_{i\in\mathcal{U}_j^\dagger}\mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*,\mathbf{x}_i+\boldsymbol{\delta}_i^*)\Big]}^{\text{Part of }\mathcal{M}_r[\mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}}]}-\Big[\sum_{i^\dagger\in\mathcal{U}^\dagger}\mathtt{D}_{\boldsymbol{\omega}\boldsymbol{\omega}} l(\boldsymbol{\omega}^*,\mathbf{x}_{i^\dagger}+\boldsymbol{\delta}_{i^\dagger}^*)\Big]\Bigg\}^{-1}\\ &\cdot\Big[\overbrace{\sum_{j=1}^r\sum_{i\in\mathcal{U}_j^\dagger}\nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*,\mathbf{x}_i+\boldsymbol{\delta}_i^*)}^{\mathcal{M}_r[\nabla_{\boldsymbol{\omega}}]}+\sum_{i^\dagger\in\mathcal{U}^\dagger}\nabla_{\boldsymbol{\omega}} l(\boldsymbol{\omega}^*,\mathbf{x}_{i^\dagger}+\boldsymbol{\delta}_{i^\dagger}^*)\Big].\end{aligned}\tag{38}$$

Then, the unlearning model with updated parameters $\boldsymbol{\omega}_{r+1}^u:=\boldsymbol{\omega}^*+\mathfrak{U}_r(\boldsymbol{\omega}^*,\mathcal{U}^\dagger)$ satisfies the approximate unlearning criteria for the adversarial training model in eq.(6).

The proof of Corollary 2 is similar to the proof of Corollary 1. Based on Corollary 2, *MUter* can be similarly designed to support the successive batch unlearning setting.