Model Sparsity Can Simplify Machine Unlearning

Jinghan Jia\(^1\), Jiancheng Liu\(^1\), Parikshit Ram\(^2\), Yuguang Yao\(^1\), Gaowen Liu\(^3\), Yang Liu\(^4,5\), Pranay Sharma\(^6\), Sijia Liu\(^1,2\)

\(^1\)Michigan State University, \(^2\)IBM Research, \(^3\)Cisco Research, \(^4\)University of California, Santa Cruz, \(^5\)ByteDance Research, \(^6\)Carnegie Mellon University

\(^*\)Equal contribution

Abstract

In response to recent data regulation requirements, machine unlearning (MU) has emerged as a critical process to remove the influence of specific examples from a given model. Although exact unlearning can be achieved through complete model retraining using the remaining dataset, the associated computational costs have driven the development of efficient, approximate unlearning techniques. Moving beyond data-centric MU approaches, our study introduces a novel model-based perspective: model sparsification via weight pruning, which is capable of reducing the gap between exact unlearning and approximate unlearning. We show in both theory and practice that model sparsity can boost the multi-criteria unlearning performance of an approximate unlearner, closing the approximation gap, while continuing to be efficient. This leads to a new MU paradigm, termed prune first, then unlearn, which infuses a sparse model prior into the unlearning process. Building on this insight, we also develop a sparsity-aware unlearning method that utilizes sparsity regularization to enhance the training process of approximate unlearning. Extensive experiments show that our proposals consistently benefit MU in various unlearning scenarios. A notable highlight is the 77% unlearning efficacy gain of fine-tuning (one of the simplest unlearning methods) when using sparsity-aware unlearning. Furthermore, we demonstrate the practical impact of our proposed MU methods in addressing other machine learning challenges, such as defending against backdoor attacks and enhancing transfer learning. Codes are available at [https://github.com/OPTML-Group/Unlearn-Sparse](https://github.com/OPTML-Group/Unlearn-Sparse).

1 Introduction

Machine unlearning (MU) initiates a reverse learning process to scrub the influence of data points from a trained machine learning (ML) model. It was introduced to avoid information leakage about private data upon completion of training [1–3], particularly in compliance with legislation like ‘the right to be forgotten’ [4] in General Data Protection Regulation (GDPR) [5]. The direct but optimal unlearning approach is exact unlearning to retrain ML models from scratch using the remaining training set, after removing the data points to be scrubbed. Although retraining yields the ground-truth unlearning strategy, it is the most computationally intensive one. Therefore, the development of approximate but fast unlearning methods has become a major focus in research [6–10].

Despite the computational benefits of approximate unlearning, it often lacks a strong guarantee on the effectiveness of unlearning, resulting in a performance gap with exact unlearning [11]. In particular, we encounter two main challenges. First, the performance of approximate unlearning can heavily rely on the configuration of algorithmic parameters. For example, the Fisher forgetting method [12] needs...
to carefully tune the Fisher information regularization parameter in each data-model setup. Second, the effectiveness of an approximate scheme can vary significantly across the different unlearning evaluation criteria, and their trade-offs are not well understood. For example, high ‘efficacy’ (ability to protect the privacy of the scrubbed data) neither implies nor precludes high ‘fidelity’ (accuracy on the remaining dataset) [9]. This raises our driving question (Q) below:

(Q) Is there a theoretically-grounded and broadly-applicable method to improve approximate unlearning across different unlearning criteria?

To address (Q), we advance MU through a fresh and novel viewpoint: model sparsification. Our key finding is that model sparsity (achieved by weight pruning) can significantly reduce the gap between approximate unlearning and exact unlearning; see Fig. 1 for the schematic overview of our proposal and highlighted empirical performance.

Model sparsification (or weight pruning) has been extensively studied in the literature [13–19], focusing on the interrelation between model compression and generalization. For example, the notable lottery ticket hypothesis (LTH) [15] demonstrated the existence of a sparse subnetwork (the so-called ‘winning ticket’) that matches or even exceeds the test accuracy of the original dense model. In addition to generalization, the impact of pruning has also been investigated on model robustness [20–22], fairness [23, 24], interpretability [25, 26], loss landscape [16, 26], and privacy [27, 28]. In particular, the privacy gains from pruning [27, 28] imply connections between data influence and model sparsification.

More recently, a few works [29, 30] attempted to draw insights from pruning for unlearning. In Wang et al. [29], removing channels of a deep neural network (DNN) showed an unlearning benefit in federated learning. And in Ye et al. [30], filter pruning was introduced in lifelong learning to detect “pruning identified exemplars” [31] that are easy to forget. However, different from the above literature that customized model pruning for a specific unlearning application, our work systematically and comprehensively explores and exploits the foundational connections between unlearning and pruning. We summarize our contributions below.

• First, we provide a holistic understanding of MU across the full training/evaluation stack.
• Second, we draw a tight connection between MU and model pruning and show in theory and practice that model sparsity helps close the gap between approximate unlearning and exact unlearning.
• Third, we develop a new MU paradigm termed ‘prune first, then unlearn’, and investigate the influence of pruning methods in the performance of unlearning. Additionally, we develop a novel ‘sparsity-aware unlearning’ framework that leverages a soft sparsity regularization scheme to enhance the approximate unlearning process.
• Finally, we perform extensive experiments across diverse datasets, models, and unlearning scenarios. Our findings consistently highlight the crucial role of model sparsity in enhancing MU.

2 Revisiting Machine Unlearning and Evaluation

Problem setup. MU aims to remove (or scrub) the influence of some targeted training data on a trained ML model [1, 2]. Let \( D = \{ z_i \}_{i=1}^N \) be a (training) dataset of \( N \) data points, with label
When scrubbing D, we next take a step further to revisit the IU method and re-derive its formula (which has received growing attention. Yet, the boosted computation efficiency comes at the cost of enhancing the effectiveness of existing solutions proposed in the previous research. The rationale is that fine-tuning on the model update from \( \theta \) to \( \theta_o \), by a scrubbing algorithm, after removing the influence of \( D_t \) from the trained model \( \theta_o \). The problem of MU is to find an accurate and efficient scrubbing mechanism to generate \( \theta_o \) from \( \theta_o \). In existing studies [2, 7, 12], the choice of the forgetting dataset \( D_t \) specifies different unlearning scenarios. There exist two main categories. First, class-wise forgetting [7, 12] refers to unlearning \( D_t \) consisting of training data points of an entire class. Second, random data forgetting corresponds to unlearning \( D_t \) given by a subset of random data drawn from all classes.

**Exact and approximate MU methods.** The exact unlearning method refers to retraining the model parameters from scratch over the remaining dataset \( D_r \). Although retraining from scratch (that we term Retrain) is optimal for MU, it entails a large computational overhead, particularly for DNN training. This problem is alleviated by approximate unlearning, an easy-to-compute proxy for Retrain, which has received growing attention. Yet, the boosted computation efficiency comes at the cost of MU’s efficacy. We next review some commonly-used approximate unlearning methods that we improve in the sequel by leveraging sparsity; see a summary in Table 1.

- **Fine-tuning (FT)** [6, 12]: Different from Retrain, FT fine-tunes the pre-trained model \( \theta \) on \( D_r \) using a few training epochs to obtain \( \theta_o \). The rationale is that fine-tuning on \( D_r \) initiates the catastrophic forgetting in the model over \( D_t \) as is common in continual learning [33].

- **Gradient ascent (GA)** [7, 8]: GA reverses the model training on \( D_t \) by adding the corresponding gradients back to \( \theta_o \), i.e., moving \( \theta_o \) in the direction of increasing loss for data points to be scrubbed.

- **Fisher forgetting (FF)** [9, 12]: FF adopts an additive Gaussian noise to ‘perturb’ \( \theta_o \) towards exact unlearning. Here the Gaussian distribution has zero mean and covariance determined by the 4th root of Fisher Information matrix with respect to (w.r.t.) \( \theta_o \) on \( D_r \). We note that the computation of the Fisher Information matrix exhibits lower parallel efficiency in contrast to other unlearning methods, resulting in higher computational time when executed on GPUs; see Golatkar et al. [12] for implementation details.

- **Influence unlearning (IU)** [10, 32]: IU leverages the influence function approach [34] to characterize the change in \( \theta_o \) as a training point is removed from the training loss. IU estimates the change in model parameters from \( \theta_o \) to \( \theta_n \), i.e., \( \theta_n = \theta_o - \theta_o \). IU also relates to an important line of research in MU, known as \( \epsilon-\delta \) forgetting [29, 35, 36]. However, it typically requires additional model and training assumptions [35].

We next take a step further to revisit the IU method and re-derive its formula (Prop. 1), with the aim of enhancing the effectiveness of existing solutions proposed in the previous research.

**Proposition 1** Given the weighted ERM training \( \theta(w) = \arg \min_{\theta} L(w, \theta) = \sum_{i=1}^{N}[w_i \ell_i(\theta, z_i)], w_i \in [0, 1] \) is the influence weight associated with the data point \( z_i \), and \( 1^T w = 1 \), the model update from \( \theta_o \) to \( \theta \) yields

\[
\Delta(w) := \theta(w) - \theta_o = H^{-1} \nabla_{\theta} L(1/N, \theta_o),
\]

where \( 1 \) is the \( N \)-dimensional vector of all ones, \( w = 1/N \) signifies the uniform weights used by ERM, \( H^{-1} \) is the inverse of the Hessian \( \nabla^2_{\theta} L(1/N, \theta_o) \) evaluated at \( \theta_o \), and \( \nabla L \) is the gradient of \( L \). When scrubbing \( D_t \), the unlearned model is given by \( \theta_u = \theta_o + \Delta(w_{MU}) \). Here \( w_{MU} \in [0, 1]^N \) with entries \( w_{MU, i} = \mathbb{1}_{D_t}(i)/|D_t| \) signifying the data influence weights for MU, \( \mathbb{1}_{D_t}(i) \) is the indicator function with value 1 if \( i \in D_t \) and 0 otherwise, and \( |D_t| \) is the cardinality of \( D_t \).

**Proof:** We derive (1) using an implicit gradient approach; see Appendix A.

It is worth noting that we have taken into consideration the weight normalization effect \( 1^T w = 1 \) in (1). This is different from existing work like Izzo et al. [10, Sec. 3] using Boolean or unbounded
Towards a ‘full-stack’ MU evaluation. Existing work has assessed MU performance from different aspects [7, 8, 12]. Yet, a single performance metric may provide a limited view of MU [11]. By carefully reviewing the prior art, we focus on the following empirical metrics (summarized in Tab. 1).

✦ Unlearning accuracy (UA): We define UA($\theta_u$) = 1 - Acc$_D$(\$\theta_u\$) to characterize the efficacy of MU in the accuracy dimension, where Acc$_D$(\$\theta_u\$) is the accuracy of \$\theta_u\$ on the forgetting dataset \$D_f\$. It is important to note that a more favorable UA for an approximate unlearning method should reduce its performance disparity with the gold-standard retrained model (Retrain); a higher value is not necessarily better. This principle also extends to other evaluation metrics.

✦ Membership inference attack (MIA) on $D_f$ (MIA-Efficacy): This is another metric to assess the efficacy of unlearning. It is achieved by applying the confidence-based MIA predictor [38, 39] to the unlearned model ($\theta_u$) on the forgetting dataset ($D_f$). The MIA success rate can then indicate how many samples in $D_f$ can be correctly predicted as forgetting (i.e., non-training) samples of $\theta_u$. A higher MIA-Efficacy implies less information about $D_f$ in $\theta_u$: see Appendix C.3 for more details.

✦ Remaining accuracy (RA): This refers to the accuracy of $\theta_u$ on $D_r$, which reflects the fidelity of MU [9], i.e., training data information should be preserved from $\theta_o$ to $\theta_u$.

✦ Testing accuracy (TA): This measures the generalization ability of $\theta_u$ on a testing dataset rather than $D_f$ and $D_r$. TA is evaluated on the whole test dataset, except for class-wise forgetting, in which testing data points belonging to the forgetting class are not in the testing scope.

✦ Run-time efficiency (RTE): This measures the computation efficiency of an MU method. For example, if we regard the run-time cost of Retrain as the baseline, the computation acceleration gained by different approximate unlearning methods is summarized in Tab. 1.

3 Model Sparsity: A Missing Factor Influencing Machine Unlearning

Model sparsification via weight pruning. Model sparsification could not only facilitate a model’s training, inference, and deployment but also benefit model’s performance. For example, LTH (lottery ticket hypothesis) [15] stated that a trainable sparse sub-model could be identified from the original dense model, with test accuracy on par or even better than the original model. Fig. 2 shows an example of the pruned model’s generalization vs. its sparsity ratio. Here one-shot magnitude pruning (OMP) [17] is adopted to obtain sparse models. OMP is computationally the lightest pruning method, which directly prunes the model weights to the target sparsity ratio based on their magnitudes. As we can see, there exists a graceful sparse regime with lossless testing accuracy.

Gains of MU from sparsity. We first analyze the impact of model sparsity on MU through a lens of unrolling stochastic gradient descent (SGD) [8]. The specified SGD method allows us to derive the unlearning error (given by the weight difference between the approximately unlearned model and the gold-standard retrained model) when scrubbing a single data point. However, different from Thudi et al. [8], we will infuse the model sparsity into SGD unrolling.

Let us assume a binary mask $m$ associated with the model parameters $\theta$, where $m_i = 0$ signifies that the $i$th parameter $\theta_i$ is pruned to zero and $m_i = 1$ represents the unmasked $\theta_i$. This sparse pattern $m$ could be obtained by a weight pruning method, like OMP. Given $m$, the sparse model is $m \odot \theta$, where $\odot$ denotes the element-wise multiplication. Thudi et al. [8] showed that if GA is adopted to scrub a single data point for the original (dense) model $\theta$ (i.e., $m = 1$), then the gap between GA and Retrain can be approximately bounded in the weight space. Prop. 2 extends the existing unlearning error analysis to a sparse model.
We next draw some key insights from Prop. 2. First, it is clear from (2) that the unlearning error reduces as the model sparsity in $\mathbf{m}$ increases. By contrast, the unlearning error derived in Thudi et al. [8] for a dense model (i.e., $\mathbf{m} = 1$) is proportional to the dense model distance $||\theta_t - \theta_0||_2$. Thus, model sparsity is beneficial to reducing the gap between (GA-based) approximate and exact unlearning. Second, the error bound (2) enables us to relate MU to the spectrum of the Hessian of the loss landscape. The number of active singular values (corresponding to nonzero dimensions in $\mathbf{m}$) decreases when the sparsity grows. However, it is important to note that in a high-sparsity regime, the model’s generalization could decrease. Consequently, it is crucial to select the model sparsity to strike a balance between generalization and unlearning performance.

Inspired by Prop. 2, we ask: Does the above benefit of model sparsification in MU apply to other approximate unlearning methods besides GA? This drives us to investigate the performance of approximate unlearning across the entire spectrum as depicted in Tab. 1. Therefore, Fig. 3 shows the unlearning efficacy (UA and MIA-Efficacy), fidelity (RA), and generalization (TA) of different approximate unlearning methods in the sparse model regime. Here class-wise forgetting is considered for MU and OMP is used for weight pruning. As we can see, the efficacy of approximate unlearning is significantly improved as the model sparsity increases, e.g., UA and MIA-Efficacy of using FT over 90% sparsity. By contrast, FT over the dense model (0% sparsity) is the least effective for MU. Also, the efficacy gap between exact unlearning (Retrain) and approximate unlearning reduces on sparse models. Further, through the fidelity and generalization lenses, FT and FF yield the RA and TA performance closest to Retrain, compared to other unlearning methods. In the regime of ultra-high sparsity (99%), the efficacy of unlearning exhibits a tradeoff with RA and TA to some extent.

### 4 Sparsity-Aided Machine Unlearning

Our study in Sec. 3 suggests the new MU paradigm ‘prune first, then unlearn’, which leverages the fact that (approximate) unlearning on a sparse model yields a smaller unlearning error (Prop. 2) and improves the efficacy of MU (Fig. 3). This promising finding, however, raises some new questions. First, it remains elusive how the choice of a weight pruning method impacts the unlearning performance. Second, it leaves room for developing sparsity-aware MU methods that can directly scrub data influence from a dense model.

#### Prune first, then unlearn: Choice of pruning methods

There exist many ways to find the desired sparse model in addition to OMP. Examples include pruning at random initialization before training [40, 41] and simultaneous pruning-training iterative magnitude pruning (IMP) [15]. Thus, the problem of pruning method selection arises for MU. From the viewpoint of MU, the unlearner would...
prioritize a pruning method that satisfies the following criteria: (1) *least dependence* on the forgetting dataset \( (D_t) \), (2) *lossless generalization* when pruning, and (3) *pruning efficiency*. The rationale behind (1) is that it is desirable not to incorporate information of \( D_t \) when seeking a sparse model prior to unlearning. And the criteria (2) and (3) ensure that sparsity cannot hamper TA (testing accuracy) and RTE (run-time efficiency). Based on (1)-(3), we propose to use two pruning methods.

**SynFlow** (synaptic flow pruning) [40]: SynFlow provides a (training-free) pruning method at initialization, even without accessing the dataset. Thus, it is uniquely suited for MU to meet the criterion (1). And SynFlow is easy to compute and yields a generalization improvement over many other pruning-at-initialization methods; see justifications in [40].

**OMP** (one-shot magnitude pruning) [17]: Different from SynFlow, OMP, which we focused on in Sec. 3, is performed over the original model \( (\theta_o, D_f) \). It may depend on the forgetting dataset \( (D_t) \), but has a much weaker dependence compared to IMP-based methods. Moreover, OMP is computationally lightest (i.e. best for (3)) and can yield better generalization than SynFlow [18].

Furthermore, it is important to clarify that IMP (iterative magnitude pruning) is *not* suitable for MU, despite being widely used to find the most accurate sparse models (i.e., best for criterion (3)). Compared with the proposed pruning methods, IMP has the largest computation overhead and the strongest correlation with the training dataset (including \( D_t \)), thereby deviating from (1) and (3). In Fig. 4, we show the efficacy of FT-based unlearning on sparse models generated using different pruning methods (SynFlow, OMP, and IMP). As we can see, unlearning on SynFlow or OMP-generated sparse models yields improved UA and MIA-Efficacy over that on the original dense model and IMP-generated sparse models. This unlearning improvement over the dense model is consistent with Fig. 3. More interestingly, we find that IMP *cannot* benefit the unlearning efficacy, although it leads to the best TA. This is because IMP heavily relies on the training set including forgetting data points, which is revealed by the empirical results – the unlearning metrics get worse for IMP with increasing sparsity. Furthermore, when examining the performance of SynFlow and OMP, we observe that the latter generally outperforms the former, exhibiting results that are closer to those of Retrain. Thus, OMP is the pruning method we will use by default.

**Sparsity-aware unlearning.** We next study if pruning and unlearning can be carried out simultaneously, without requiring prior knowledge of model sparsity. Let \( L_u(\theta; \theta_o, D_f) \) denote the unlearning objective function of model parameters \( \theta \), given the pre-trained state \( \theta_o \), and the remaining training dataset \( D_f \). Inspired by sparsity-inducing optimization [42], we integrate an \( \ell_1 \)-norm-based sparse penalty into \( L_u \). This leads to the problem of \( \ell_1 \)-sparse MU:

\[
\theta_u = \arg \min_{\theta} L_u(\theta; \theta_o, D_f) + \gamma \| \theta \|_1, \tag{3}
\]

where we specify \( L_u \) by the fine-tuning objective, and \( \gamma > 0 \) is a regularization parameter that controls the penalty level of the \( \ell_1 \) norm, thereby reducing the magnitudes of 'unimportant' weights.

Table 2: MU performance comparison of using \( \ell_1 \)-sparse MU with different sparsity schedulers of \( \gamma \) in (3) and using Retrain. The unlearning scenario is given by random data forgetting (10% data points across all classes) on (ResNet-18, CIFAR-10). A performance gap against Retrain is provided in (●).

In practice, the unlearning performance could be sensitive to the choice of the sparse regularization parameter \( \gamma \). To address this limitation, we propose the design of a sparse regularization scheduler. Specifically, we explore three schemes: (1) constant \( \gamma \), (2) linearly growing \( \gamma \) and (3) linearly decaying \( \gamma \); see Sec. 5.1 for detailed implementations. Our empirical evaluation presented in Tab. 2
shows that the use of a linearly decreasing $\gamma$ scheduler outperforms other schemes. This scheduler not only minimizes the gap in unlearning efficacy compared to Retrain, but also improves the preservation of RA and TA after unlearning. These findings suggest that it is advantageous to prioritize promoting sparsity during the early stages of unlearning and then gradually shift the focus towards enhancing fine-tuning accuracy on the remaining dataset $D_7$.

5 Experiments

5.1 Experiment setups

Datasets and models. Unless specified otherwise, our experiments will focus on image classification under CIFAR-10 [43] using ResNet-18 [44]. Yet, experiments on additional datasets (CIFAR-100 [43], SVHN [45], and ImageNet [46]) and an alternative model architecture (VGG-16 [47]) can be found in Appendix C.4. Across all the aforementioned datasets and model architectures, our experiments consistently show that model sparsification can effectively reduce the gap between approximate unlearning and exact unlearning.

Unlearning and pruning setups. We focus on two unlearning scenarios mentioned in Sec. 2, class-wise forgetting and random data forgetting (10% of the whole training dataset together with 10 random trials). In the ‘prune first, then unlearn’ paradigm, we focus on unlearning methods (FT, GA, FF, and IU) shown in Tab. 1 when applying to sparse models. We implement these methods following their official repositories. However, it is worth noting that the implementation of FF in Golatkar et al. [12] modifies the model architecture in class-wise forgetting, i.e., removes the prediction head of the class to be scrubbed. By contrast, other methods keep the model architecture intact during unlearning. Also, we choose OMP as the default pruning method, as justified in Fig. 4. In the ‘sparsity-aware unlearning’ paradigm, the sparsity-promoting regularization parameter $\gamma$ in (3) is determined through the line search in the interval $[10^{-5}, 10^{-1}]$, with consideration for the trade-off between testing accuracy and unlearning accuracy. For all schedulers, $\gamma$ is set around $5 \times 10^{-4}$. The linearly increasing and decaying schedulers are implemented as $\gamma_t = \frac{\gamma_1}{T} \gamma$ and $\gamma_t = \frac{1}{t} \gamma$ respectively, where $t$ is the epoch index and $T$ is the total number of epochs. We refer readers to Appendix C.2 for more details.

Evaluation metrics. We evaluate the unlearning performance following Tab. 1. Recall that UA and MIA-Efficacy depict the efficacy of MU, RA reflects the fidelity of MU, and TA and RETE characterize the generalization ability and the computation efficiency of an unlearning method. We implement MIA (membership inference attack) using the prediction confidence-based attack method [38, 39], whose effectiveness has been justified in Song and Mittal [48] compared to other methods. We refer readers to Appendix C.3 for more implementation details. To more precisely gauge the proximity of each approximate MU to Retrain, we introduce a metric termed ‘Disparity Average’. This metric quantifies the mean performance gap between each unlearning method and Retrain across all considered metrics. A lower value indicates closer performance to Retrain.

5.2 Experiment results

Table 3: Performance overview of various MU methods on dense and 95%-sparse models considering different unlearning scenarios: class-wise forgetting, and random data forgetting. The forgetting data of random data forgetting ratio is 10% of the whole training dataset, the sparse models are obtained using OMP [17], and the unlearning methods and evaluation metrics are summarized in Tab. 1. Class-wise forgetting is conducted class-wise. The performance is reported in the form $a \pm b$, with mean $a$ and standard deviation $b$ computed over 10 independent trials. A performance gap against Retrain is provided in (●). Note that the better performance of approximate unlearning corresponds to the smaller performance gap with the gold-standard retrained model. ‘Disparity Ave.’ represents the average unlearning gaps across diverse metrics.
Model sparsity improves approximate unlearning. In Tab. 3, we study the impact of model sparsity on the performance of various MU methods in the ‘prune first, then unlearn’ paradigm. The performance of the exact unlearning method (Retrain) is also provided for comparison. Note that the better performance of approximate unlearning corresponds to the smaller performance gap with the gold-standard retrained model.

First, given an approximate unlearning method (FT, GA, FF, or IU), we consistently observe that model sparsity improves UA and MIA-Efficacy (i.e., the efficacy of approximate unlearning) without much performance loss in RA (i.e., fidelity). In particular, the performance gap between each approximate unlearning method and Retrain reduces as the model becomes sparser (see the ‘95% sparsity’ column vs. the ‘dense’ column). Note that the performance gap against Retrain is highlighted in (·) for each approximate unlearning. We also observe that Retrain on the 95%-sparsity model encounters a 3% TA drop. Yet, from the perspective of approximate unlearning, this drop brings in a more significant improvement in UA and MIA-Efficacy when model sparsity is promoted. Let us take FT (the simplest unlearning method) for class-wise forgetting as an example. As the model sparsity reaches 95%, we obtain 51% UA improvement and 8% MIA-Efficacy improvement. Furthermore, FT and IU on the 95%-sparsity model can better preserve TA compared to other methods. Table 3 further indicates that sparsity reduces average disparity compared to a dense model across various approximate MU methods and unlearning scenarios.

Second, existing approximate unlearning methods have different pros and cons. Let us focus on the regime of 95% sparsity. We observe that FT typically yields the best RA and TA, which has a tradeoff with its unlearning efficacy (UA and MIA-Efficacy). Moreover, GA yields the worst RA since it is most loosely connected with the remaining dataset D_s. FF becomes ineffective when scrubbing random data points compared to its class-wise unlearning performance. Furthermore, IU causes a TA drop but yields the smallest gap with exact unlearning across diverse metrics under the 95% model sparsity. In Appendix C.4, we provide additional results on CIFAR-100 and SVHN datasets, as shown in Tab. A3, as well as on the ImageNet dataset, depicted in Tab. A5. Other results pertaining to the VGG-16 architecture are provided in Tab. A4.

Effectiveness of sparsity-aware unlearning. In Fig. 5, we showcase the effectiveness of the proposed sparsity-aware unlearning method, i.e., ℓ_1-sparse MU. For ease of presentation, we focus on the comparison with FT and the optimal Retrain strategy in both class-wise forgetting and random data forgetting scenarios under (CIFAR-10, ResNet-18). As we can see, ℓ_1-sparse MU outperforms FT in the unlearning efficacy (UA and MIA-Efficacy), and closes the performance gap with Retrain without losing the computation advantage of approximate unlearning. We refer readers to Appendix C.4 and Fig. A2 for further exploration of ℓ_1-sparse MU on additional datasets.

Application: MU for Trojan model cleanse. We next present an application of MU to remove the influence of poisoned backdoor data from a learned model, following the backdoor attack setup [49], where an adversary manipulates a small portion of training data (a.k.a. poisoning ratio) by injecting a backdoor trigger (e.g., a small image patch) and modifying data labels towards a targeted incorrect label. The trained model is called Trojan model, yielding the backdoor-designated incorrect prediction if the trigger is present at testing. Figure 6: Performance of Trojan model cleanse via proposed unlearning vs. model sparsity, where ‘Original’ refers to the original Trojan model. Left: ASR vs. model sparsity. Right: SA vs. model sparsity. Otherwise, it behaves normally.
We then regard MU as a defensive method to scrub the harmful influence of poisoned training data in the model’s prediction, with a similar motivation as Liu et al. [50]. We evaluate the performance of the unlearned model from two perspectives, backdoor attack success rate (ASR) and standard accuracy (SA). Fig. 6 shows ASR and SA of the Trojan model (with poisoning ratio 10%) and its unlearned version using the simplest FT method against model sparsity. Fig. 6 also includes the $\ell_1$-sparse MU to demonstrate its effectiveness on model cleanse. Since it is applied to a dense model (without using hard thresholding to force weight sparsity), it contributes just a single data point at the sparsity level 0%. As we can see, the original Trojan model maintains 100% ASR and a similar SA across different model sparsity levels. By contrast, FT-based unlearning can reduce ASR without inducing much SA loss. Such a defensive advantage becomes more significant when sparsity reaches 90%. Besides, $\ell_1$-sparse MU can also effectively remove the backdoor effect while largely preserving the model’s generalization. Thus, our proposed unlearning shows promise in application of backdoor attack defense.

**Application: MU to improve transfer learning.** Further, we utilize the $\ell_1$-sparse MU method to mitigate the impact of harmful data classes of ImageNet on transfer learning. This approach is inspired by Jain et al. [51], which shows that removing specific negatively-influenced ImageNet classes and retraining a source model can enhance its transfer learning accuracy on downstream datasets after finetuning. However, retraining the source model introduces additional computational overhead. MU naturally addresses this limitation and offers a solution.

Tab. 4 illustrates the transfer learning accuracy of the unlearned or retrained source model (ResNet-18) on ImageNet, with $n$ classes removed. The downstream target datasets used for evaluation are SUN397 [52] and OxfordPets [53]. The employed finetuning approach is linear probing, which finetunes the classification head of the source model on target datasets while keeping the feature extraction network of the source model intact. As we can see, removing data classes from the source ImageNet dataset can lead to improved transfer learning accuracy compared to the conventional method of using the pre-trained model on the full ImageNet ($i.e., n = 0$). Moreover, our proposed $\ell_1$-sparse MU method achieves comparable or even slightly better transfer learning accuracy than the retraining-based approach in Jain et al. [51]. Importantly, $\ell_1$-sparse MU offers the advantage of computational efficiency $2 \times$ speed up over Jain et al. [51] across all cases, making it an appealing choice for transfer learning using large-scale models. Here we remark that in order to align with Jain et al. [51], we employed a fast-forward computer vision training pipeline (FFCV) [54] to accelerate our ImageNet training on GPUs.

**Additional results.** We found that model sparsity also enhances the privacy of the unlearned model, as evidenced by a lower MIA-Privacy. Refer to Appendix C.4 and Fig. A1 for more results. In addition, we have expanded our experimental scope to encompass the ‘prune first, then unlearn’ approach across various datasets and architectures. The results can be found in Tab. A3, Tab. A4, and Tab. A5.

Furthermore, we conducted experiments on the $\ell_1$-sparse MU across different datasets, the Swin-Transformer architecture, and varying model sizes within the ResNet family. The corresponding findings are presented in Fig. A2 and Tab. A6, A7, A8 and A9.

### 6 Related Work

While Sec. 2 provides a summary of related works concerning exact and approximate unlearning methods and metrics, a more comprehensive review is provided below.

**Machine unlearning.** In addition to exact and approximate unlearning methods as we have reviewed in Sec. 2, there exists other literature aiming to develop the probabilistic notion of unlearning [35, 55–58], in particular through the lens of differential privacy (DP) [59]. Although DP enables unlearning with provable error guarantees, they typically require strong model and algorithmic assumptions and could lack effectiveness when facing practical adversaries, e.g., membership inference attacks. Indeed, evaluating MU is far from trivial [8, 9, 11]. Furthermore, the attention on MU has also been raised.
in different learning paradigms, e.g., federated learning [29, 60], graph neural networks [61–63], and adversarial ML [64, 65]. In addition to preventing the leakage of data privacy from the trained models, the concept of MU has also inspired other emergent applications such as adversarial defense against backdoor attacks [6, 50] that we have studied and erasing image concepts of conditional generative models [66, 67].

Understanding data influence. The majority of MU studies are motivated by data privacy. Yet, they also closely relate to another line of research on understanding data influence in ML. For example, the influence function approach [32] has been used as an algorithmic backbone of many unlearning methods [6, 10]. From the viewpoint of data influence, MU has been used in the use case of adversarial defense against data poisoning backdoor attacks [50]. Beyond unlearning, evaluation of data influence has also been studied in fair learning [68, 69], transfer learning [51], and dataset pruning [70, 71].

Model pruning. The deployment constraints on e.g., computation, energy, and memory necessitate the pruning of today’s ML models, i.e., promoting their weight sparsity. The vast majority of existing works [13–19] focus on developing model pruning methods that can strike a graceful balance between model’s generalization and sparsity. In particular, the existence of LTH (lottery ticket hypothesis) [15] demonstrates the feasibility of co-improving the model’s generalization and efficiency (in terms of sparsity) [40, 72–75]. In addition to generalization, model sparsity achieved by pruning can also be leveraged to improve other performance metrics, such as robustness [20–22], model explanation [25, 26], and privacy [27, 28, 76, 77].

7 Conclusion

In this work, we advance the method of machine unlearning through a novel viewpoint: model sparsification, achieved by weight pruning. We show in both theory and practice that model sparsity plays a foundational and crucial role in closing the gap between exact unlearning and existing approximate unlearning methods. Inspired by that, we propose two new unlearning paradigms, ‘prune first, then unlearn’ and ‘sparsity-aware unlearn’, which can significantly improve the efficacy of approximate unlearning. We demonstrate the effectiveness of our findings and proposals in extensive experiments across different unlearning setups. Our study also indicates the presence of model modularity traits, such as weight sparsity, that could simplify the process of machine unlearning. This may open up exciting prospects for future research to investigate unlearning patterns within weight or architecture space.

8 Acknowledgement

The work of J. Jia, J. Liu, Y. Yao, and S. Liu were supported by the Cisco Research Award and partially supported by the NSF Grant IIS-2207052. Y. Liu was partially supported by NSF Grant IIS-2143895 and IIS-2040800.

References


Appendix

A Proof of Proposition 1

Recap the definition of model update $\Delta(w)$ in (1) and $\theta_o = \theta(1/N)$, we approximate $\Delta(w)$ by the first-order Taylor expansion of $\theta(w)$ at $w = 1/N$. This leads to

$$\Delta(w) = \theta(w) - \theta(1/N) \approx \frac{d\theta(w)}{dw}\bigg|_{w=1/N} (w - 1/N), \quad (A1)$$

where $\frac{d\theta(w)}{dw} \in \mathbb{R}^{M \times N}$, and recall that $M = |\theta_o|$ is the number of model parameters. The gradient $\frac{d\theta(w)}{dw}$ is known as implicit gradient [78] since it is defined through the solution of the optimization problem $\theta(w) = \arg \min_{\theta} L(w, \theta)$, where recall that $L(w, \theta) = \sum_{i=1}^{N} [w_i \ell_i(\theta, z_i)]$. By the stationary condition of $\theta(w)$, we obtain

$$\nabla_{\theta} L(w, \theta(w)) = 0. \quad (A2)$$

Next, we take the derivative of (A2) w.r.t. $w$ based on the implicit function theorem [78] assuming that $\theta(w)$ is the unique solution to minimizing $L$. This leads to

$$\left[\frac{d\theta(w)}{dw}\right]^{T} \left[\nabla_{\theta} \theta L(w, \theta)\right]_{\theta=\theta(w)} + \nabla_{w, \theta} L(w, \theta(w)) = 0, \quad (A3)$$

where $\nabla_{a,b} = \nabla_{a} \nabla_{b} \in \mathbb{R}^{[a] \times [b]}$ is the second-order partial derivative. Therefore,

$$\frac{d\theta(w)}{dw} = - \left[\nabla_{\theta} \theta L(w, \theta(w))\right]^{-1} \nabla_{w, \theta} L(w, \theta(w))^{T}, \quad (A4)$$

where $\nabla_{w, \theta} L(w, \theta(w))$ can be expanded as

$$\nabla_{w, \theta} L(w, \theta(w)) = \nabla_{w} \nabla_{\theta} \sum_{i=1}^{N} [w_i \ell_i(\theta(w), z_i)] \quad (A5)$$

$$= \nabla_{w} \sum_{i=1}^{N} [w_i \nabla_{\theta} \ell_i(\theta(w), z_i)] \quad (A6)$$

$$= \begin{bmatrix}
\nabla_{\theta}^{1} \ell_1(\theta, z_1)^{T} \\
\nabla_{\theta}^{1} \ell_2(\theta, z_2)^{T} \\
\vdots \\
\nabla_{\theta}^{1} \ell_N(\theta, z_N)^{T}
\end{bmatrix} \quad (A7)$$

Based on (A4) and (A7), we obtain the closed-form of implicit gradient at $w = 1/N$:

$$\frac{d\theta(w)}{dw}\bigg|_{w=1/N} = - \left[\nabla_{\theta} \theta L(1/N, \theta(1/N))\right]^{-1} \left[\nabla_{\theta} \ell_1(\theta(1/N), z_1) \ldots \nabla_{\theta} \ell_N(\theta(1/N), z_N)\right]$$

$$= - H^{-1} \left[\nabla_{\theta} \ell_1(\theta(1/N), z_1) \ldots \nabla_{\theta} \ell_N(\theta(1/N), z_N)\right], \quad (A8)$$

where $H = \nabla_{\theta} \theta L(1/N, \theta(1/N))$.

Substituting (A8) into (A1), we obtain

$$\Delta(w) \approx - H^{-1} \left[\nabla_{\theta} \ell_1(\theta(1/N), z_1) \ldots \nabla_{\theta} \ell_N(\theta(1/N), z_N)\right] (w - 1/N)$$

$$= - H^{-1} \sum_{i=1}^{N} [(w_i - 1/N) \nabla_{\theta} \ell_i(\theta(1/N), z_i)]$$

$$= H^{-1} \nabla_{\theta} L(1/N - w, \theta_o), \quad (A9)$$

where the last equality holds by the definition of $L(w, \theta) = \sum_{i=1}^{N} [w_i \ell_i(\theta, z_i)]$.

The proof is now complete. \qed
Remark on IU using ave-ERM vs. sum-ERM. Recall that the weighted empirical risk minimization (ERM) loss used in proposition (1), \( L(w, \theta) = \sum_{i=1}^{N} w_i \ell_i(\theta, z_i) \) corresponds to the ave-ERM as \( w \) is subject to the simplex constraint \( (I^T w = 1 \text{ and } w \geq 0) \). This is different from the conventional derivation of IU using the sum-ERM \([32, 35]\) in the absence of simplex constraint. In what follows, we discuss the impact of ave-ERM on IU vs. sum-ERM.

Noting that \( \theta_o \) represents the original model trained through conventional ERM, namely, the weighted ERM loss by setting \( w_i = c (\forall i) \) for a positive constant \( c \). Given the unlearning scheme (encoded in \( w \)), the IU approach aims to delineate the model parameter adjustments required by MU from \( \theta_o \) mentioned in proposition (1). The difference between ave-ERM and sum-ERM would play a role in deriving \( \Delta(w) \), which relies on the Taylor expansion of \( \theta(w) \) (viewed as a function of \( w \)). When the sum-ERM is considered, then the linearization point is typically set by \( w = 1 \). This leads to

\[
\Delta^{(\text{sum})}(w) = \theta(w) - \theta(1) \approx \theta(1) + \frac{d\theta(w)}{dw}|_{w=1} (w - 1) - \theta(1)
\]

where \( \theta(1) = \theta_o \) for sum-ERM, and \( \frac{d\theta(w)}{dw} \) is implicit gradient \([78]\) since it is defined upon an implicit optimization problem \( \theta(w) = \arg\min_{\theta} L(w, \theta) \).

When the ave-ERM is considered, the linearization point is set by \( w = 1/N \). This leads to

\[
\Delta^{(\text{ave})}(w) = \theta(w) - \theta(1/N) \approx \frac{d\theta(w)}{dN}|_{w=1/N} (w - 1/N) - \theta(1/N)
\]

where \( \theta(1/N) = \theta_o \) for ave-ERM. The derivation of the implicit gradient \( \frac{d\theta(w)}{dw} \) is shown in (A8).

Next, let us draw a comparison between \( \Delta^{(\text{sum})}(w) \) and \( \Delta^{(\text{ave})}(w) \) using a specific example below. If we aim to unlearn the first \( k \) training data points, the unlearning weights \( w_{MU} \) under sum-ERM is then given by \( w_{MU}^{(\text{sum})} = [0, 0, \ldots, 0, 1, 1, \ldots, 1] \), where 0 encodes the data sample to be unlearned or removed. This yields \( (w_{MU}^{(\text{sum})} - 1) = [1, 1, \ldots, 1, 0, 0, \ldots, 0] \). By contrast, the unlearning weights \( w_{MU}^{(\text{ave})} \) under ave-ERM is given by \( w_{MU}^{(\text{ave})} = [0, 0, \ldots, 0, \frac{1}{N-k}, \frac{1}{N-k}, \ldots, \frac{1}{N-k}] \). As a result,

\[
(w_{MU}^{(\text{ave})} - 1/N) = \underbrace{[\frac{1}{N}, \frac{1}{N}, \ldots, \frac{1}{N}, \frac{1}{N-k}, \frac{1}{N-k}, \ldots, \frac{1}{N-k}]}_{k \neq s}
\]

is caused by the presence of simplex constraint of \( w \) in ave-ERM. Thus, the MU’s weight configuration \( (w_{MU}^{(\text{ave})} - 1/N) \) obtained from ave-ERM is different from \( (w_{MU}^{(\text{sum})} - 1) \) in the sum-ERM setting.

Given the above example, the error term of the Taylor expansion using sum-ERM for \( w = w_{MU}^{\text{sum}} \) is in the order of \( ||w_{MU}^{\text{sum}} - 1||^2 \), while the error term using ave-ERM for \( w = w_{MU}^{\text{ave}} \) is in the order of \( ||w_{MU}^{\text{ave}} - 1/N||^2 = \frac{k^2}{N(N-k)} \). Thus compared to ave-ERM, the use of sum-ERM could cause the first-order Taylor expansion in IU less accurate as the number of unlearning datapoints \( (k) \) increases. Furthermore, the IG \( \frac{d\theta(w)}{dw}|_{w=1/N} \) in sum-ERM is also different from \( \frac{d\theta(w)}{dw}|_{w=1/N} \) in ave-ERM as they are evaluated at two different linearization points.

B Proof of Proposition 2

The proof follows \([8, \text{Sec. 5}]\), with the additional condition that the model is \textbf{sparse} encoded by a pre-fixed (binary) pruning mask \( m \), namely, \( \theta' := m \odot \theta \). Then, based on \([8, \text{Eq. 5}]\), the model
updated by SGD yields

\[
\theta'_i \approx \theta'_0 - \eta m \odot \sum_{i=1}^{t-1} \nabla_{\theta} \ell(\theta'_0, \hat{z}_i) + m \odot \left( \sum_{i=1}^{t-1} f(i) \right),
\]  

(A12)

where \( \theta'_0 = m \odot \theta_0 \) is the model initialization when using SGD-based sparse training, \( \{ \hat{z}_i \} \) is the sequence of stochastic data samples, \( t \) is the number of training iterations, \( \eta \) is the learning rate, and \( f(i) \) is defined recursively as

\[
f(i) = -\eta \nabla^2_{\theta, \theta} \ell(\theta'_0, \hat{z}_i) \left( -\eta \sum_{j=0}^{i-1} m \odot \nabla_{\theta} \ell(\theta'_0, \hat{z}_j) + \sum_{j=0}^{i-1} (m \odot f(j)) \right),
\]

(A13)

with \( f(0) = 0 \). Inspired by the second term of (A12), to unlearn the data sample \( \hat{z}_i \), we will have to add back the first-order gradients under \( \hat{z}_i \). This corresponds to the GA-based approximate unlearning method. Yet, this approximate unlearning introduces an unlearning error, given by the last term of (A12)

\[
e_m(\theta_0, \{ \hat{z}_i \}, t, \eta) := m \odot \left( \sum_{i=1}^{t-1} f(i) \right).
\]

(A14)

Next, if we interpret the mask \( m \) as a diagonal matrix \( \text{diag}(m) \) with 0’s and 1’s along its diagonal based on \( m \), we can then express the sparse model \( m \odot \theta \) as \( \text{diag}(m) \theta \). Similar to [8, Eq. 9], we can derive a bound on the unlearning error (A14) by ignoring the terms other than those with \( \eta^2 \) in \( f(i) \), \textit{i.e.}, (A13). This is because, in the recursive form of \( f(i) \), all other terms exhibit a higher degree of the learning rate \( \eta \) compared to \( \eta^2 \). As a result, we obtain

\[
e(m) = \| e_m(\theta_0, \{ \hat{z}_i \}, t, \eta) \|_2 = \left\| m \odot \left( \sum_{i=1}^{t-1} f(i) \right) \right\|_2  
\approx \eta^2 \left\| \text{diag}(m) \sum_{i=1}^{t-1} \nabla^2_{\theta, \theta} \ell(\theta'_0, \hat{z}_i) \sum_{j=0}^{i-1} m \odot \nabla_{\theta} \ell(\theta'_0, \hat{z}_j) \right\|_2  
\leq \eta^2 \sum_{i=1}^{t-1} \left\| \text{diag}(m) \nabla^2_{\theta, \theta} \ell(\theta'_0, \hat{z}_i) \sum_{j=0}^{i-1} m \odot \nabla_{\theta} \ell(\theta'_0, \hat{z}_j) \right\|_2 \quad \text{(Triangle inequality)}
\leq \eta^2 \sum_{i=1}^{t-1} \| \text{diag}(m) \nabla^2_{\theta, \theta} \ell(\theta'_0, \hat{z}_i) \|_2 \left\| \sum_{j=0}^{i-1} m \odot \nabla_{\theta} \ell(\theta'_0, \hat{z}_j) \right\|_2 \quad \text{(A15)}
\leq \eta^2 \sum_{i=1}^{t-1} \| \text{diag}(m) \nabla^2_{\theta, \theta} \ell(\theta'_0, \hat{z}_i) \|_2 \| \sigma(m) \|_2  
\leq \eta^2 \sigma(m) \left\| m \odot (\theta_t - \theta_0) \right\|_2 \frac{1}{t} t - 1 - t = \frac{\eta^2}{2} (t - 1) \left\| m \odot (\theta_t - \theta_0) \right\|_2 \sigma(m),
\]

(A17)

where the inequality (A16) holds given the fact that \( \sum_{j=0}^{i-1} m \odot \nabla_{\theta} \ell(\theta'_0, \hat{z}_j) \) in (A15) can be approximated by its expectation \( i(\theta'_0 - \theta_0) \) [8, Eq. 7], and \( \sigma(m) := \max_j \{ \sigma_j(\nabla^2_{\theta, \theta} \ell), \text{if } m_j \neq 0 \} \), \textit{i.e.}, the largest eigenvalue among the dimensions left intact by the binary mask \( m \). The above suggests that the unlearning error might be large if \( m = 1 \) (no pruning). Based on (A17), we can then readily obtain the big \( O \) notation in (2). This completes the proof.

### C Additional Experimental Details and Results

#### C.1 Datasets and models

We summarize the datasets and model configurations in Tab. A1.
Table A1: Dataset and model setups.

<table>
<thead>
<tr>
<th>Settings</th>
<th>CIFAR-10</th>
<th>SVHN</th>
<th>CIFAR-100</th>
<th>ImageNet</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ResNet-18</td>
<td>VGG-16</td>
<td>ResNet-18</td>
<td>ResNet-18</td>
</tr>
<tr>
<td>Batch Size</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>128</td>
</tr>
</tbody>
</table>

Table A2: Detailed training details for model pruning.

<table>
<thead>
<tr>
<th>Experiments</th>
<th>CIFAR-10/CIFAR-100</th>
<th>SVHN</th>
<th>ImageNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training epochs</td>
<td>182</td>
<td>160</td>
<td>90</td>
</tr>
<tr>
<td>Rewinding epochs</td>
<td>8</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>Momentum</td>
<td>0.9</td>
<td>0.9</td>
<td>0.875</td>
</tr>
<tr>
<td>(\ell_2) regularization</td>
<td>(5e^{-4})</td>
<td>(5e^{-4})</td>
<td>(3.05e^{-5})</td>
</tr>
<tr>
<td>Warm-up epochs</td>
<td>1 (75 for VGG-16)</td>
<td>0</td>
<td>8</td>
</tr>
</tbody>
</table>

C.2 Additional training and unlearning settings

**Training configuration of pruning.** For all pruning methods, including IMP [15], SynFlow [40], and OMP [17], we adopt the settings from the current SOTA implementations [17]; see a summary in Tab. A2. For IMP, OMP, and SynFlow, we adopt the step learning rate scheduler with a decay rate of 0.1 at 50\% and 75\% epochs. We adopt 0.1 as the initial learning rate for all pruning methods.

**Additional training details of MU.** For all datasets and model architectures, we adopt 10 epochs for FT, and 5 epochs for GA method. The learning rate for FT and GA are carefully tuned between \([10^{-5}, 0.1]\) for each dataset and model architecture. In particular, we adopt 0.01 as the learning rate for FT method and \(10^{-4}\) for GA on the CIFAR-10 dataset (ResNet-18) at different sparsity levels. By default, we choose SGD as the optimizer for the FT and GA methods. As for FF method, we perform a greedy search for hyperparameter tuning [12] between \(10^{-8}\) and \(10^{-6}\).

C.3 Detailed metric settings

**Details of MIA implementation.** MIA is implemented using the prediction confidence-based attack method [48]. There are mainly two phases during its computation: (1) **training phase**, and (2) **testing phase**. To train an MIA model, we first sample a balanced dataset from the remaining dataset \(D_r\) and the test dataset (different from the forgetting dataset \(D_f\)) to train the MIA predictor. The learned MIA is then used for MU evaluation in its testing phase. To evaluate the performance of MU, MIA-Efficacy is obtained by applying the learned MIA predictor to the unlearned model \((\theta_u)\) on the forgetting dataset \(D_f\). Our objective is to find out how many samples in \(D_f\) can be correctly predicted as non-training samples by the MIA model against \(\theta_u\). The formal definition of MIA-Efficacy is then given by:

\[
\text{MIA-Efficacy} = \frac{TN}{|D_f|},
\]

where \(TN\) refers to the true negatives predicted by our MIA predictor, i.e., the number of the forgetting samples predicted as non-training examples, and \(|D_f|\) refers to the size of the forgetting dataset. As described above, MIA-Efficacy leverages the privacy attack to justify how good the unlearning performance could be.

C.4 Additional experiment results

**Model sparsity benefits privacy of MU for ‘free’.** It was recently shown in [27, 28] that model sparsification helps protect data privacy, in terms of defense against MIA used to infer training data information from a learned model. Inspired by the above, we ask if sparsity can also bring the privacy benefit to an unlearned model, evaluated by the MIA rate on the remaining dataset \(D_r\) (that we term MIA-Privacy). This is different from MIA-Efficacy, which reflects the efficacy of scrubbing \(D_f\), i.e., correctly predicting that data sample in \(D_f\) is not in the training set of the unlearned model. In contrast, MIA-Privacy characterizes the privacy of the unlearned model about \(D_f\). A lower MIA-Privacy implies less information leakage.
Fig. A1 shows MIA-Privacy of unlearned models versus the sparsity ratio applied to different unlearning methods in the ‘prune first, then unlearn’ paradigm. As we can see, MIA-Privacy decreases as the sparsity increases. This suggests the improved privacy of unlearning on sparse models. Moreover, we observe that approximate unlearning outperforms exact unlearning (Retrain) in privacy preservation of $D_F$. This is because Retrain is conducted over $D_F$ from scratch, leading to the strongest dependence on $D_F$ than other unlearning methods. Another interesting observation is that IU and GA yield a much smaller MIA-Privacy than other approximate unlearning methods. The rationale behind that is IU and GA have a weaker correlation with $D_F$ during unlearning. Specifically, the unlearning loss of IU only involves the forgetting data influence weights, i.e., $(1/N - w)$ in (1). Similarly, GA only performs gradient ascent over $D_F$, with the least dependence on $D_F$.

Performance of ‘prune first, then unlearn’ on various datasets and architectures. As demonstrated in Tab. A3 and Tab. A4, the introduction of model sparsity can effectively reduce the discrepancy between approximate and exact unlearning across a diverse range of datasets and architectures. This phenomenon is observable in various unlearning scenarios. Remarkably, model sparsity enhances both UA and MIA-Efficacy metrics without incurring substantial degradation on RA and TA in different unlearning scenarios. These observations corroborate the findings reported in Tab. 3.

To demonstrate the effectiveness of our methods on a larger dataset, we conducted additional experiments on ImageNet [46] with settings consistent with the class-wise forgetting in Tab. 3. As we can see from Tab. A5, sparsity reduces the performance gap between exact unlearning (Retrain) and the approximate unlearning methods (FT and GA). The results are consistent with our observations in other datasets. Note that the 83% model sparsity (ImageNet, ResNet-18) is used to preserve the TA performance after one-shot magnitude (OMP) pruning.

Performance of $\ell_1$ sparsity-aware MU on additional datasets. As seen in Fig. A2, $\ell_1$-sparse MU significantly reduces the gap between approximate and exact unlearning methods across various datasets.
datasets (CIFAR-100 [43], SVHN [45], ImageNet [46]) in different unlearning scenarios. It notably outperforms other methods in UA and MIA-Efficacy metrics while preserving acceptable RA and TA performances, thus becoming a practical choice for unlearning scenarios. In class-wise and random data forgetting cases, $\ell_1$-sparse MU exhibits performance on par with Retrain in UA and MIA-Efficacy metrics. Importantly, the use of $\ell_1$-sparse MU consistently enhances forgetting metrics with an insignificant rise in computational cost compared with FT, underscoring its effectiveness and efficiency in diverse unlearning scenarios. For detailed numerical results, please refer to Tab. A6.

![Figure A2: Performance of sparsify-aware unlearning vs. FT and Retrain on class-wise forgetting and random data forgetting under (CIFAR-10, ResNet-18). Each metric is normalized to [0, 1] based on the best result across unlearning methods for ease of visualization, while the actual best value is provided. The figure format is consistent with Fig. 5.](image)

Table A6: Performance of sparsity-aware MU vs. Retrain, FT, GA and IU considering class-wise forgetting and random data forgetting, where the table format and setup are consistent with Tab. 3. The unit of RTE is minutes for all datasets, except ImageNet. For ImageNet, indicated by an asterisk (*), RTE is measured by hours.

![Table A6: Performance of MU vs. sparsity on ImageNet considering class-wise forgetting. The content format follows Tab. 3.](image)
Table A7: Performance of $\ell_1$-sparsely MU vs. Retrain and FT on (Swin Transformer, CIFAR-10).

<table>
<thead>
<tr>
<th>MU</th>
<th>UA</th>
<th>MAE-Frac</th>
<th>RA</th>
<th>ITA</th>
<th>RTE (n/a)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class-wise forgetting</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Retrain</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>80.14</td>
<td>151.00</td>
</tr>
<tr>
<td>$\ell_1$-sparsely MU</td>
<td>5.65 (81.94)</td>
<td>22.46 (77.54)</td>
<td>98.92 (0.06)</td>
<td>79.72 (0.42)</td>
<td>9.89</td>
</tr>
<tr>
<td>Random data forgetting</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Retrain</td>
<td>81.65</td>
<td>80.00</td>
<td>78.50</td>
<td>69.00</td>
<td>154.00</td>
</tr>
<tr>
<td>FT</td>
<td>80.46 (81.32)</td>
<td>8.12 (77.18)</td>
<td>79.80 (0.05)</td>
<td>79.54 (0.05)</td>
<td>7.77</td>
</tr>
<tr>
<td>$\ell_1$-sparsely MU</td>
<td>5.22 (82.06)</td>
<td>16.51 (81.14)</td>
<td>79.22 (0.06)</td>
<td>79.59 (0.05)</td>
<td>7.84</td>
</tr>
</tbody>
</table>

Performance of $\ell_1$ sparsity-aware MU on additional architectures. Tab. A7 presents an additional application to Swin Transformer on CIFAR-10. To facilitate a comparison between approximate unlearning methods (including the FT baseline and the proposed $\ell_1$-sparsely MU) and Retrain, we train the transformer from scratch on CIFAR-10. This could potentially decrease testing accuracy compared with fine-tuning on a pre-trained model over a larger, pre-trained dataset. As we can see, our proposed $\ell_1$-sparsely MU leads to a much smaller performance gap with Retrain compared to FT. In particular, class-wise forgetting exhibited a remarkable 90.24% increase in UA, accompanied by a slight reduction in RA.

Performance of ‘prune first, then unlearn’ and $\ell_1$ sparsity-aware MU on different model sizes. Further, Tab A8 and Tab A9 present the unlearning performance versus different model sizes in the ResNet family, involving both ResNet-20s and ResNet-50 on CIFAR-10, in addition to ResNet-18 in Tab. 3. As we can see, sparsity consistently diminishes the unlearning gap with Retrain (indicated by highlighted numbers, with smaller values being preferable). It’s worth noting that while both ResNet-20s and ResNet-50 benefit from sparsity, the suggested sparsity ratio is 90% for ResNet-20s and slightly lower than 95% for ResNet-50 when striking the balance between MU and generalization.

Table A8: MU performance on (ResNet-20s, CIFAR-10) using ‘prune first, then unlearn’ (applying to the OMP-resulted 90% sparse model) and ‘sparse-aware unlearning’ (applying to the original dense model). The performance is reported in the form $d_{a,b}$, with mean $a$ and standard deviation $b$ computed over 10 independent trials. A performance gap against Retrain is provided in (●). Smaller performance gap from Retrain is better in the context of machine unlearning.

<table>
<thead>
<tr>
<th>MU</th>
<th>Dense</th>
<th>UA</th>
<th>90% Sparsity</th>
<th>Dense</th>
<th>MAE-Frac</th>
<th>RA</th>
<th>ITA</th>
<th>RTE (n/a)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class-wise forgetting</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Retrain</td>
<td>63.00 ± (10.89)</td>
<td>94.00 ± (8.31)</td>
<td>100.00 ± (5.24)</td>
<td>94.75 ± (4.36)</td>
<td>94.75 ± (4.36)</td>
<td>96.81 ± (4.36)</td>
<td>96.81 ± (4.36)</td>
<td>95.87</td>
</tr>
<tr>
<td>$\ell_1$-sparsely MU</td>
<td>60.99 ± (11.83)</td>
<td>93.50 ± (9.05)</td>
<td>97.17 ± (7.83)</td>
<td>99.27 ± (6.01)</td>
<td>96.14 ± (4.82)</td>
<td>93.11 ± (0.38)</td>
<td>94.00 ± (4.36)</td>
<td>94.00 ± (4.36)</td>
</tr>
<tr>
<td>Random data forgetting</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Retrain</td>
<td>60.14 ± (11.83)</td>
<td>93.50 ± (9.05)</td>
<td>97.17 ± (7.83)</td>
<td>99.27 ± (6.01)</td>
<td>96.14 ± (4.82)</td>
<td>93.11 ± (0.38)</td>
<td>94.00 ± (4.36)</td>
<td>94.00 ± (4.36)</td>
</tr>
<tr>
<td>$\ell_1$-sparsely MU</td>
<td>60.14 ± (11.83)</td>
<td>93.50 ± (9.05)</td>
<td>97.17 ± (7.83)</td>
<td>99.27 ± (6.01)</td>
<td>96.14 ± (4.82)</td>
<td>93.11 ± (0.38)</td>
<td>94.00 ± (4.36)</td>
<td>94.00 ± (4.36)</td>
</tr>
</tbody>
</table>

Table A9: MU performance on (ResNet-50, CIFAR-10) following the format of Table A8.

<table>
<thead>
<tr>
<th>MU</th>
<th>Dense</th>
<th>UA</th>
<th>90% Sparsity</th>
<th>Dense</th>
<th>MAE-Frac</th>
<th>RA</th>
<th>ITA</th>
<th>RTE (n/a)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class-wise forgetting</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Retrain</td>
<td>70.90 ± (14.20)</td>
<td>97.84 ± (4.21)</td>
<td>100.00 ± (5.10)</td>
<td>100.00 ± (5.10)</td>
<td>94.10 ± (4.36)</td>
<td>96.72 ± (4.36)</td>
<td>96.72 ± (4.36)</td>
<td>95.20</td>
</tr>
<tr>
<td>$\ell_1$-sparsely MU</td>
<td>60.44 ± (6.35)</td>
<td>94.00 ± (4.36)</td>
<td>97.17 ± (7.83)</td>
<td>99.27 ± (6.01)</td>
<td>96.14 ± (4.82)</td>
<td>93.11 ± (0.38)</td>
<td>94.00 ± (4.36)</td>
<td>94.00 ± (4.36)</td>
</tr>
</tbody>
</table>

D Broaden Impacts and Limitations

Broader Impacts. Our study on model sparsity-inspired MU provides a versatile solution to forget arbitrary data points and could give a general solution for dealing with different concerns, such as the model's privacy, efficiency, and robustness. Moreover, the applicability of our method extends beyond these aspects, with potential impacts in the following areas. ① Regulatory compliance: Our method enables industries, such as healthcare and finance, to adhere to regulations that require the forgetting of data after a specified period. This capability ensures compliance while preserving the utility and performance of machine learning models. ② Fairness: Our method could also play a crucial role in addressing fairness concerns by facilitating the unlearning of biased datasets or subsets. By removing biased information from the training data, our method contributes to mitigating bias in machine
learning models, ultimately fostering the development of fairer models. \(\text{③ ML with adaptation and sustainability:}\) Our method could promote the dynamic adaptation of machine learning models by enabling the unlearning of outdated information, and thus, could enhance the accuracy and relevance of the models to the evolving trends and dynamics of the target domain. This capability fosters sustainability by ensuring that ML models remain up-to-date and adaptable, thus enabling their continued usefulness and effectiveness over time.

**Limitations.** One potential limitation of our study is the absence of provable guarantees for \(\ell_1\)-sparse MU. Since model sparsification is integrated with model training as a soft regularization, the lack of formal proof may raise concerns about the reliability and robustness of the approach. Furthermore, while our proposed unlearning framework is generic, its applications have mainly focused on solving computer vision tasks. As a result, its effectiveness in the domain of natural language processing (NLP) remains unverified. This consideration becomes particularly relevant when considering large language models. Therefore, further investigation is necessary for future studies to explore the applicability and performance of the framework in NLP tasks.