Improved Localized Machine Unlearning Through the Lens of Memorization

Reihaneh Torkzadehmahani[*†] *reihaneh.torkzadehmahani@tum.de*
Technical University of Munich

Reza Nasirigerdeh *reza.nasirigerdeh@tum.de*
Technical University of Munich

Georgios Kaissis *g.kaissis@gmail.com*
Google DeepMind
Technical University of Munich

Daniel Rueckert *daniel.rueckert@tum.de*
Technical University of Munich
Imperial College London

Gintare Karolina Dziugaite *gkdz@google.com*
Google DeepMind
Mila

Eleni Triantafillou *etriantafillou@google.com*
Google DeepMind

Reviewed on OpenReview: *https://openreview.net/forum?id=zXAVdHYPIB*

Abstract

Machine unlearning refers to removing the influence of a specified subset of training data from a model efficiently, after it has already been trained. This is important for key applications, including making the model more accurate by removing outdated, mislabeled, or poisoned data. In this paper, we draw inspiration from prior work that attempts to identify where in the network a given example is memorized, to propose a new "localized unlearning" algorithm, Deletion by Example Localization (DEL). DEL has two components: a localization strategy that identifies critical parameters for a given set of examples, and a simple unlearning algorithm that finetunes only the critical parameters on the data we want to retain. Through extensive experiments, we find that our localization strategy outperforms prior strategies in terms of metrics of interest for unlearning and test accuracy, and pairs well with various unlearning algorithms. Our experiments on different datasets, forget sets, and metrics reveal that DEL outperforms prior work in producing better trade-offs between unlearning performance and accuracy [1].

1 Introduction

Machine unlearning, coined by Cao & Yang (2015), is the problem of removing from a trained model (the influence of) a subset of its original training dataset. While unlearning is a young area of research, it has

[*]Corresponding author.
[†]Work done during the author's research internship at Google DeepMind.
[1]The code is available at: https://github.com/reihaneh-torkzadehmahani/DEL-Unlearning/

recently attracted a lot of attention (Triantafillou et al., 2024). Example applications of unlearning include keeping models up-to-date or improving their quality by deleting training data that is identified post-training as being outdated, mislabeled or poisoned.

Unlearning is a challenging problem in deep neural networks since they are highly non-convex, preventing us from easily quantifying the influence of different training examples on the trained weights. As a straightforward solution to unlearning a given "forget set", one can retrain the model from scratch excluding that set. This approach implements *exact unlearning*, guaranteeing that the resulting model has no influence from the forget set. However, this can be prohibitively computationally expensive. Instead, a burgeoning area of research has emerged that designs methods to post-process the trained model to attempt to *approximately* erase the influence of the forget set *efficiently*. This introduces a challenging balancing act, as imperfect attempts at removing some training examples after the fact may accidentally damage the model and overly reduce its utility (e.g., accuracy on the remainder of the training data or generalization ability). Therefore, designing successful approximate unlearning methods involves navigating trade-offs between i) unlearning quality, ii) utility, and iii) efficiency.

We hypothesize that *localized unlearning*, where the unlearning algorithm operates only on a (small) subset of parameters, is promising for striking a good balance in the above trade-offs. Specifically, modifying a (appropriately chosen) small fraction of the weights may intuitively be less likely to overly damage the network's utility and more likely to be efficient, since fewer parameters are subject to modification. However, the success of such a localized approach hinges on the ability to identify the right subset of parameters to perform unlearning on. In this work, we take a deep dive into different localization strategies, drawing inspiration specifically from hypotheses formulated about *where* in the network training data is "memorized".[2]

Indeed, a closely-related research community has been studying *memorization* in neural networks. Informally, a training example is memorized by a model if that model's predictions on that example would have been different had the example not been included in the training set (Feldman, 2020; Pruthi et al., 2020). As we discuss later, this notion is closely tied to unlearning.

Our contributions can be summarized as follows:

- We leverage hypotheses for where examples are memorized to derive localization strategies for unlearning and investigate their strengths and weaknesses. We find that data-agnostic strategies perform poorly: they either achieve good unlearning performance at the expense of utility, or the other way around, but not both.
- We propose a practical localization strategy inspired by the memorization localization algorithm of Maini et al. (2023). Our strategy is more efficient than that algorithm and, when paired with various unlearning algorithms from the literature, outperforms prior work in terms of unlearning and utility metrics.
- We propose a new localized unlearning algorithm, Deletion by Example Localization (DEL), by pairing our localization strategy with the simple approach of resetting the deemed-to-be critical parameters and then finetuning the newly reinitialized parameters.
- DEL performs strongly on three datasets/architectures (CIFAR-10 with ResNet-18, SVHN with ViT, and ImageNet-100 with ResNet-50), different unlearning settings (subclass unlearning, class unlearning, and continual unlearning), and different forget sets (IID and non-IID in subclass unlearning), against prior localized and full-parameter unlearning algorithms. Across the board, DEL is the best performer (or no worse) in terms of forgetting quality, while almost always outperforming all localized methods in terms of test accuracy too.
- In addition, DEL is more robust to the parameter budget, outperforming the previous state-of-the-art method SalUn across different budgets.

[2]We intend here a very restricted definition of "memorization" in the context of classification models, which do not provide generative outputs. In this context, memorization relates to a data element contributing to the model's ability to accurately label input data. Such models do not "contain" bit-wise or code-wise copies of their training data.

2 Background

We begin by introducing notation and defining key concepts.

Let $\mathcal{D}_{\text{train}}$ denote a training dataset and $\mathcal{A}$ a (possibly randomized) training algorithm. We denote by $\theta^o = \mathcal{A}(\mathcal{D}_{\text{train}})$ the parameters obtained by training on $\mathcal{D}_{\text{train}}$ using $\mathcal{A}$. We refer to θ^o as the "original model", i.e. before unlearning takes place. We will study algorithms for "unlearning" a subset $\mathcal{S} \subset \mathcal{D}_{\text{train}}$, referred to as the forget set. We refer to $\mathcal{D}_{\text{train}} \setminus \mathcal{S}$, the remaining training data, as the retain set.

2.1 Unlearning

We define unlearning intuitively in a way that reflects the standard metrics used in the community that we also adopt for evaluation; see Section A.1 for additional discussion.

Definition 2.1. Unlearning. For a given algorithm $\mathcal{A}$ and dataset $\mathcal{D}_{\text{train}}$, an algorithm $\mathcal{U}$ is said to unlearn a forget set $\mathcal{S}$ if the unlearned model $\mathcal{U}(\theta^o, \mathcal{S}, \mathcal{D}_{\text{train}} \setminus \mathcal{S})$ and the "retrained model" $\mathcal{A}(\mathcal{D}_{\text{train}} \setminus \mathcal{S})$ have the same distribution of outputs on $\mathcal{S}$.

The above compares the (distribution of) outputs of the models obtained by two different recipes. The first recipe is $\mathcal{A}(\mathcal{D}_{\text{train}} \setminus \mathcal{S})$, retraining "from scratch" on only the retain set, which is prohibitively expensive but ideal from the standpoint of eliminating the influence of $\mathcal{S}$ on the model. The second is $\mathcal{U}(\theta^o, \mathcal{S}, \mathcal{D}_{\text{train}} \setminus \mathcal{S})$, applying $\mathcal{U}$ to post-process the original model θ^o in order to unlearn $\mathcal{S}$. Note that, while different variations are possible, we assume that $\mathcal{U}$ has access to both the forget set and the retain set.

In the above, we refer to "distributions" of outputs since re-running either recipe with a different random seed, which controls, e.g., the initialization or the order of mini-batches, would yield slightly different model weights, thus possibly slightly different outputs too. In our experiments considering classification tasks, the "outputs" are the vector of softmax probabilities, and different metrics consider different elements of that vector, e.g., the accuracy metric requires the argmax of that vector, whereas more sophisticated metrics consider the correct class probability (confidence).

A successful unlearning algorithms $\mathcal{U}$ is one that causes these two recipes to yield similar outputs, with the second recipe being substantially more computationally-efficient compared to the first, in order to justify paying the cost of approximate unlearning rather than simply retraining.

Unlearning evaluation. Evaluating unlearning rigorously is an ongoing area of research; current state-of-the-art evaluation methods (Hayes et al., 2024; Triantafillou et al., 2024) require training a large number of models, which is very expensive. In this work, we leverage standard metrics in the research community, building on top of the evaluation procedure of Fan et al. (2023) that considers two metrics for unlearning quality. The first is the accuracy of the unlearned model on the forget set, with the goal of matching the accuracy of the retrained model on the forget set, in line with Definition 2.1. The second is a Membership Inference Attack (MIA), which, given access to outputs ("predictions") of the unlearned model, aims to detect whether an example was used in training. Moreover, in Section A.5, we present evaluations using a stronger MIA that is based on the "confidence" of the unlearned model. We adopt the $\text{MIA}_{\text{score}}$ of Fan et al. (2023)[3] that measures the efficacy of defending against such an attack, as the portion of forget set examples that the attacker thinks were unseen. An ideal unlearning algorithm would have a $\text{MIA}_{\text{score}}$ matching that of the retrained-from-scratch model; see Section A.4 for details. In addition, a comprehensive evaluation of unlearning also requires measuring utility, which we measure via test accuracy (and retain accuracy, in Section A.6), and efficiency.

In addition to the accuracy, we employ a more fine-grained evaluation metric, where we compute the percentage of forget set samples that the oracle model misclassifies but the unlearned model classifies correctly. This metric is a better indicator of unlearning effectiveness than accuracy. This if oracle and unlearned models misclassify completely different subsets of the samples, and the size of the subsets are equal, the accuracy will report the same value for both oracle and unlearned models. The unlearning algorithm in this case is not

[3]Departing from the terminology of Fan et al. (2023), we refer to this metric as the "MIA score" instead of "MIA efficacy" to clarify that the metric measures the effectiveness of *defending against* the attack rather than the effectiveness of the attack itself.

effective in fact, but accuracy values indicate otherwise, which is misleading. The new metric does not suffer from this problem, implying that it better represents the performance of unlearning. Please see section A.8 for more details and evaluation results.

For compactness in tables, we sometimes report a difference Δ obtained by subtracting the unlearned model's value for a given metric from the oracle's value for that metric, e.g. Δ_{forget} is the forget set accuracy of the oracle model (retrained from scratch) minus that of the unlearned model. For each such Δ, the lower the absolute value, the better.

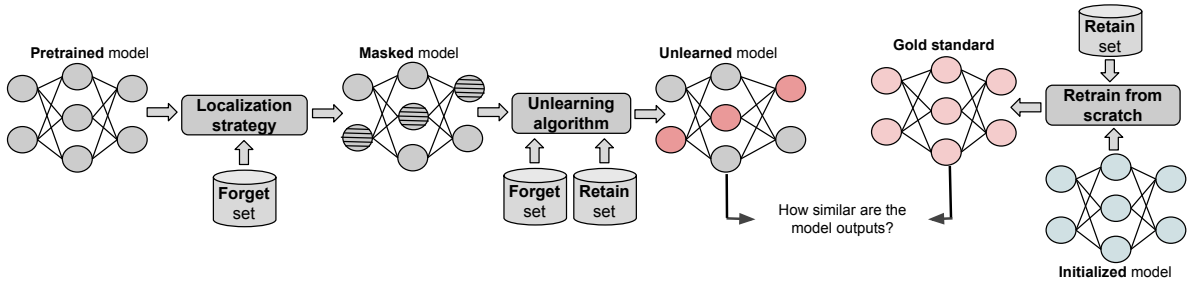


Figure 1: **Localized unlearning** consists of two parts: a *localization strategy* that identifies a set of "critical parameters" (dashed line circles) and an *unlearning algorithm* that aims to remove the influence of the forget set by modifying only the critical parameters (highlighted circles), keeping the rest unchanged. Ideally, the unlearned model should "behave" like the model retrained from scratch (oracle model), i.e., the two should produce the same (distribution of) outputs; see Definition 2.1

Localized unlearning. We focus on "localized unlearning" algorithms that modify only a (preferably small) subset of the parameters of θ^o, leaving the rest unchanged (see Figure 1). We view this as a promising direction as we hypothesize that it can yield better trade-offs between unlearning efficacy, utility, and efficiency, due to modifying fewer parameters. Indeed, the current state-of-the-art unlearning algorithm is a localized one (Fan et al., 2023).

A localized unlearning algorithm has two components. The first is a *localization strategy* $\mathcal{L}$ that produces a mask m determining which subset of the parameters should be modified to carry out the unlearning request: $m = \mathcal{L}(\theta^o, \mathcal{S})$ where m is a binary vector specifying whether each parameter will be updated. The second component is a *unlearning strategy*. This, in principle, can be any unlearning algorithm that, in this case, will operate on only the subset of the parameters indexed by m. Overall, a localized unlearning algorithm is instantiated via a $(\mathcal{L},\mathcal{U})$ pair.

2.2 Memorization

An intriguing phenomenon is that, despite models exhibiting strong generalization properties, they still tend to "memorize" some of their training data (Arpit et al., 2017; Zhang et al., 2021). In fact, recent theories argue that some forms of memorization are actually necessary for optimal generalization (Feldman, 2020; Brown et al., 2021; Attias et al., 2024). In the below, we first present a definition of memorization borrowed from (Feldman, 2020), and then discuss the connections with unlearning.

Definition 2.2. Label memorization. Assume a dataset $\mathcal{D}_{\text{train}} = \{(x_i, y_i)\}_{i=1}^N$, where x_i and y_i denote the input and label for the i'th training example, and assume a training algorithm $\mathcal{A}$ and a model $f(x;\theta)$ parameterized by θ, mapping inputs to labels. Then, the *memorization score* for an example $(x_i, y_i) \in \mathcal{D}_{\text{train}}$ (with respect to $\mathcal{D}_{\text{train}}$, $\mathcal{A}$ and f) is

$$\Pr_{\theta\sim\mathcal{A}(\mathcal{D}_{\text{train}})}[f(x_i;\theta) = y_i] \; - \Pr_{\theta\sim\mathcal{A}(\mathcal{D}_{\text{train}}\setminus(x_i,y_i))}[f(x_i;\theta) = y_i]. \tag{1}$$

Intuitively, an example is highly memorized if the model can only predict its label correctly when that example is in the training set. This will be primarily the case for atypical, ambiguous or mislabeled examples that would not be otherwise correctly predicted (Feldman & Zhang, 2020).

Connections with unlearning. Some forms of memorization and unlearning are intimately connected: at the extreme where an example isn't memorized at all, it can be thought of as being trivially "unlearned" according to some unlearning metrics of interest, because the model's predictions on that example aren't different from what they would have been had that example not been included in the training set ("retrain from scratch"). Empirically, Zhao et al. (2024) showed that most approximate unlearning methods are more successful on forget sets that include examples that have lower memorization scores compared to those with higher memorization scores. Relatedly, Jagielski et al. (2022) study catastrophic forgetting during training; a phenomenon that can be characterized as reduced memorization of an example in later stages of training, that can be interpreted as a passive form of unlearning. They find that, when training on large datasets, examples that were only seen early in training may be less memorized, which they quantify via the failure rates of privacy attacks aiming to extract examples or infer whether they were used for training. Toneva et al. (2018) find that examples with noisy labels witness a larger number of "forgetting events" during training, defined as an event where an example that was previously correctly predicted becomes incorrectly predicted later in training.

Based on these insights on the strong connection of memorization and unlearning, we ask: does knowledge (or assumptions) of *where* in the network a forget set is memorized give rise to improved unlearning for that forget set?

Localizing memorization. Investigating the above question is a challenging undertaking, as pinpointing where memorization occurs is in and of itself a research problem. Baldock et al. (2021) define the "prediction depth" for an example to be the earliest layer in the network after which the example is correctly predicted. They find that mislabeled examples are only predicted correctly in the final few layers of the model, and conclude that "early layers generalize while later layers memorize". Stephenson et al. (2021) draw the same conclusion through a study of manifold complexity and shattering capability. However, Maini et al. (2023) found that the parameters that memorize specific examples are actually scattered throughout the network and not concentrated in any individual layer.

3 Related Work

Unlearning. Cao & Yang (2015) coined the term unlearning and proposed exact algorithms for statistical query learning. Bourtoule et al. (2021); Yan et al. (2022) propose frameworks that support exact unlearning in deep networks more efficiently by considering architectures with many components, where one only needs to retrain the affected parts of the model for each unlearning request. However, in the worst case, efficiency can be as poor as in naive retraining, and furthermore, these specialized architectures may have lower accuracy compared to state-of-the-art ones. Instead, a plethora of approximate unlearning methods (Ginart et al., 2019; Guo et al., 2019; Golatkar et al., 2020a;b; Thudi et al., 2022; Seo et al., 2025) were developed that operate on an already-trained model to remove the influence of the forget set. Simple commonly-used baselines include finetuning the model on only the retain set ("**Finetune**"), or on only the forget set using the negated gradient ("**NegGrad**"), or combining these two in a joint optimization "**NegGrad+**" (Kurmanji et al., 2024). One could also apply a joint optimization using gradient descent on both the retain and forget sets, after having first randomly relabelled the examples in the forget set ("**Random Label**") (Graves et al., 2021; Fan et al., 2023). **SCRUB** (Kurmanji et al., 2024) builds on NegGrad+ by casting unlearning as a teacher-student problem and using distillation. Liu et al. (2024) show that sparsity aids unlearning and that adding an L1-penalty to the Finetune baseline improves its performance ("**L1-sparse**"). One could also utilize influence function analysis (Koh & Liang, 2017) to mitigate the impact of the forget data by estimating the importance of the model weights ("**IU**") (Izzo et al., 2021).

Localized unlearning. Goel et al. (2022) propose baselines that perform unlearning on only the k deepest (closest to the output) layers and either simply finetune them ("**CF-k**") or reinitialize them and then finetune them ("**EU-k**"). Foster et al. (2024) uses the Fisher information matrix to identify parameters that are disproportionately important to the forget set and apply unlearning on those ("**SSD**"). Fan et al. (2023) proposes Saliency Unlearning ("**SalUn**"), which selects a subset of critical parameters by considering the gradients of the forget set and applies unlearning (by default using Random Label) on the identified parameters. We will later compare this state-of-the-art method to other strategies inspired by the memorization literature.

Related ideas have been proposed in the context of unlearning in LLMs (Hase et al., 2024; Guo et al., 2024), though the problem formulation of LLM unlearning as well as the architectures and algorithms used there are substantially different and out of the scope of this paper; see discussion in Section 7.

4 Localization via memorization hypotheses

In this section, we derive localization strategies from hypotheses in the memorization literature and investigate their performance in the context of localized unlearning compared to existing approaches. Based on Section 2.2, we consider two hypotheses for where memorization occurs, and we then turn each one of them into a localization strategy $\mathcal{L}(\theta^o, \mathcal{S})$. The two hypotheses are: i) memorization happens in the "deepest layers", i.e. closest to the output, consistent with (Baldock et al., 2021; Stephenson et al., 2021), and ii) memorization of an example is confined to a small set of channels, scattered across the network, whose location depends on the example (Maini et al., 2023).

We compare the unlearning performance of four localization strategies: i) **Deepest** and ii) **CritMem**, derived based on the above two hypotheses, respectively, iii) selecting the shallowest layers ("**Shallowest**"), as a control experiment for i, and iv) the localization strategy of SalUn (Fan et al., 2023) ("**SalLoc**") that is not motivated through the lens of memorization but is regarded as state-of-the-art in the unlearning literature. Note that we reserve the name "SalUn" for the pairing of SalLoc with the Random Label unlearning algorithm, which is the choice that Fan et al. (2023) found worked best (we mix-and-match localization strategies with unlearning algorithms in Section 5).

Deepest returns a mask that allows only the k deepest (closest to the output) layers to be updated during unlearning (first hypothesis). If paired with an unlearning algorithm that resets the chosen parameters and then finetunes them, this corresponds to the EU-k method of (Goel et al., 2022).

CritMem is based on the second hypothesis and implemented via the algorithm of Maini et al. (2023). Given a training example, it iteratively finds the deemed-to-be most critical channel, resets its associated parameters and repeats, until the prediction on that example flips to an incorrect one. At each iteration, the criticality of a channel for an example is determined by multiplying its weights by their gradients on that example, and then taking the sum (over the individual channel parameters) of the magnitude of this quantity. We run this algorithm for each example in $\mathcal{S}$ independently, record the critical channels identified for each, take their union, and define a mask that turns on all parameters associated with the channels in that union.

Shallowest returns a mask that allows only the k shallowest (closest to the input) layers to be updated during unlearning, as a control experiment for Deepest.

SalLoc (Fan et al., 2023) sorts all parameters based on the magnitude of gradients over $\mathcal{S}$, in descending order. Then, given a threshold α for the percentage of parameters that may be updated during unlearning, it returns a mask that turns on only the first α percent of elements of that list.

The above strategies differ along several axes. Firstly, **Deepest** and **Shallowest** are data-agnostic in that they do not use $\mathcal{S}$ to inform which parameters to choose, whereas **CritMem** and **SalLoc** are data-dependent and tailor the mask to $\mathcal{S}$. Secondly, there are differences in granularity: **Deepest** and **Shallowest** consider a layer as the unit (i.e. each layer is either included or excluded as a whole). **CritMem**'s unit is a channel while **SalLoc** has the finest granularity, considering each individual parameter as a unit. Finally, **CritMem** is substantially more computationally expensive than **SalLoc**. This is because *for each* example in $\mathcal{S}$, it applies an *iterative* approach that resets the next most critical channel, one at a time, until the termination criterion. On the other hand, **SalLoc** chooses the critical parameters in "one-shot" rather than iteratively and, additionally, operates on *batches* of $\mathcal{S}$.

Experimental setup We conduct this investigation on CIFAR-10 using ResNet-18, and a forget set comprised of 10% of the training samples, randomly selected from two classes (2 and 5). We use a simple unlearning algorithm that resets the identified critical parameters (according to the given localization strategy), and then finetunes only those parameters and the classifier layer using the retain set ("**Reset + Finetune**"). Since each localization strategy chooses a different set and number of parameters, we compare them to one another for different "budgets" of how many parameters are updated. For example, for Deepest, we select the k

deepest layers (for several different values of k), and we compute the number of parameters that are selected each time to obtain the "budget". We report three metrics: Forget accuracy and MIA score for unlearning quality, and test accuracy for utility. For each metric, the ideal behaviour is to match the performance of the retrain-from-scratch "oracle" model. We tune hyperparameters separately for each budget and localization strategy; see details in Section A.3.

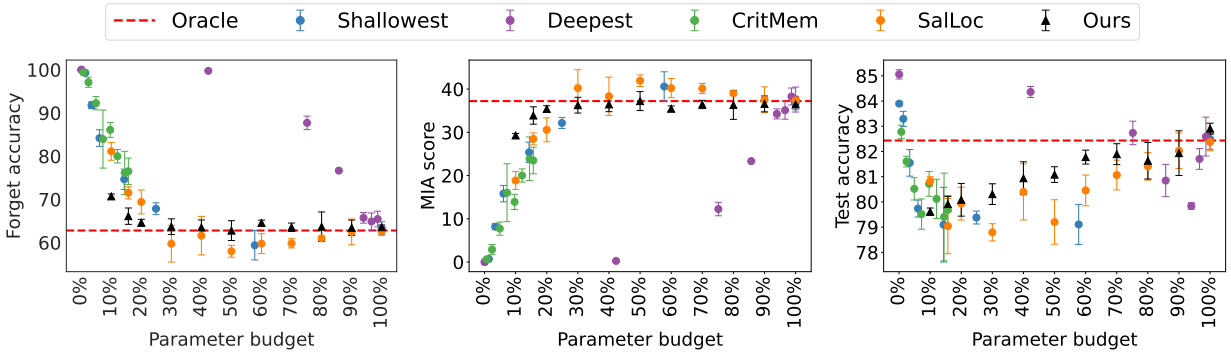


Figure 2: Comparison of localization strategies combined with the Reset + Finetune (RFT) unlearning algorithm. An ideal unlearning algorithm would match the "oracle" ("retrain-from-scratch") on each metric, with the smallest possible parameter budget, for increased efficiency. The strategy we will propose later ("ours") yields the best trade-off, with near-perfect unlearning for several budgets.

Findings From Figure 2 we observe the following. First, when updating (almost) all parameters, **Deepest** achieves strong results on all metrics, which is expected since, at that end of the spectrum, Deepest paired with Reset + Finetune amounts to retraining (almost) the entire model. However, for smaller budgets, it performs very poorly in unlearning metrics as resetting only deeper layers does not cause a sufficient accuracy drop on the forget set (though it at least preserves test accuracy). On the other hand, **Shallowest** is much more effective at unlearning compared to Deepest, perhaps due to resetting earlier layers causing larger "disruption" to the information flow in the network. Indeed, contrary to Deepest, this strategy leads to poor test accuracy for several budgets, which is the main downside of this approach. We hope that future research considers this baseline alongside EU-k of Goel et al. (2022). **CritMem** is unable to reach good unlearning quality in this setup. Note that we can only evaluate CritMem for small budgets, as the algorithm of Maini et al. (2023) terminates once enough parameters are reset such that the prediction is flipped to an incorrect one (so the max value of α we can consider is capped). We find that resetting only those parameters is insufficient for achieving good unlearning results when paired with this unlearning algorithm. **SalLoc** has a similar trend to Shallowest in terms of unlearning metrics but leads to higher test accuracy. It performs similarly to CritMem in the range where we can compare them.

Building on these findings, we also compare the computational times of four localization strategies (Shallowest, CritMem, SalLoc, and DEL) across three different forget sets of varying sizes. As shown in Table 17 in the Appendix, CritMem is the most computationally expensive strategy, whereas Shallowest (or Deepest) is the least demanding. Both SalLoc and DELlocalization strategies exhibit comparable computational times, requiring only a few seconds to execute.

Overall, we observe that data-agnostic localization strategies perform poorly: each of Deepest and Shallowest can either achieve good forgetting quality at the expense of utility, or the other way around, but not both. We hypothesize that a data-dependent approach with finer-grained control of *which* shallow or deep parameters to update in a way that is informed by the forget set, would yield better trade-offs, due to causing minimal and targeted "disruption" that preserves utility. We have considered two such data-dependent strategies so far, CritMem and SalLoc. CritMem, directly borrowed from Maini et al. 2023, results in updating only a small number of parameters (due to its termination criterion) and is unable to achieve good forgetting quality within that budget (and in the context of this unlearning algorithm), while it is also computationally

Table 1: Combining different granularity and criticality criteria in a non-iterative localization algorithm. The leftmost cell corresponds to SalLoc, and the rightmost cell corresponds to our proposed approach. Each Δ is the difference, in the specified metric, between the unlearned and retrained models (smaller is better).

	Parameter		**Channel**	
	Grads	Weighted Grads	Grads	Weighted Grads
$\Delta_{\text{Forget}} \downarrow$	$-7.58_{\pm 8e-3}$	$15.07_{\pm 0.01}$	$-12.33_{\pm 8e-3}$	$\mathbf{1.58_{\pm 0.01}}$
$\Delta_{\text{MIA}} \downarrow$	$8.74_{\pm 1.10}$	$-16.32_{\pm 1.39}$	$15.53_{\pm 1.10}$	$\mathbf{3.33_{\pm 1.28}}$
$\Delta_{\text{Test}} \downarrow$	$3.63_{\pm 4e-3}$	$11.55_{\pm 6e-3}$	$\mathbf{1.55_{\pm 2e-3}}$	$4.41_{\pm 2e-3}$

expensive, as discussed above. SalLoc, on the other hand, is more efficient and causes a smaller test accuracy drop compared to Shallowest.

5 Improved localized unlearning

In this section, building on our previous observations, we take a deeper dive into key design choices of CritMem and SalLoc and formulate and investigate hypotheses about their suitability for localized unlearning. Our findings give rise to an improved unlearning algorithm.

5.1 Dissecting building blocks of localization

Granularity. Because attempts to locate where memorization occurs rely on heuristics and are imperfect, it is easier to succeed in making coarse-grained decisions (e.g. there is a critical parameter somewhere within a given channel) rather than individual parameter-level assessments. Then, could it be that coarser granularity (as in CritMem) provides useful "smoothing" to identify critical 'regions' and is less error-prone than finer granularity (as in SalLoc), though potentially at the cost of selecting more parameters than strictly necessary?

Criticality criteria. We hypothesize that using weighted gradients (as in CritMem) rather than simply considering the magnitude of gradients themselves (as in SalLoc) is also more suitable for localized unlearning. Intuitively, a weight with a small value can be seen as less critical overall (for the training set) which is an important additional signal to consider in addition to the gradients on specific forget set examples, as it may also serve as an additional useful "regularizer" when making heuristic assessments.

Table 1 investigates the effect of the granularity and criticality criteria mentioned above, in the context of batched and non-iterative localization algorithms, on the same experimental setup (dataset, forget set, etc) as in Section 4. We find that indeed the best choice is given by using the channel-wise granularity and weighted gradients.

While Table 1 reveals that CritMem's granularity and criticality criterion are more suitable than SalLoc's for localized unlearning, recall that CritMem is computationally expensive: for each given example, it determines the next most critical channel *one at a time*, resets it and repeats to find the next most critical one, until the label of the example flips. In the next section, we propose an approach that incorporates CritMem's granularity and its criticality criterion into an efficient algorithm that, like SalLoc, estimates the critical parameters for each batch in $\mathcal{S}$ in "one-shot", and additionally operates on *batches* of examples in $\mathcal{S}$.

5.2 Introducing our localization strategy and DEL

Given a forget set $\mathcal{S}$ and a model θ^o with p parameters, for $j \in \{1, \ldots, p\}$, let θ^o_j and $g_j(\theta^o, \mathcal{S})$ represent the weight and gradient values on the forget set, respectively, for the parameter at index j. We define the *criticality score* s_j of the parameter at index j as the magnitude of the weighted gradient over the forget set: $s_j = |\theta^o_j \cdot g_j(\theta^o, \mathcal{S})|$; this is the same criticality criterion used in CritMem, whereas SalLoc simply considers the magnitude of the gradient for each parameter $|g_j(\theta^o, \mathcal{S})|$ for assessing its criticality.

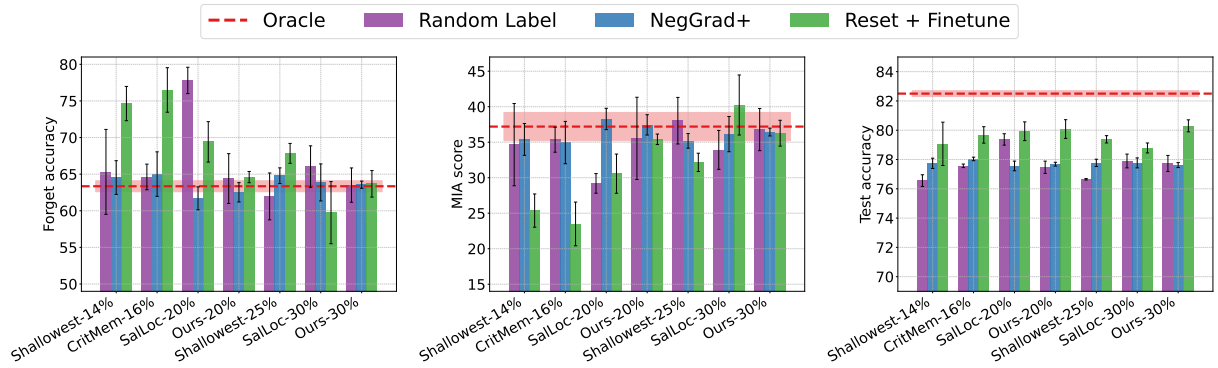


Figure 3: Pairing localization strategies / budgets (e.g. Ours-30% denotes applying our localization strategy to select 30% of parameters) with three unlearning algorithms, on CIFAR-10 / ResNet (the ideal behaviour is to match the "Oracle"). **Ours has the best unlearning efficacy, paired with *any* unlearning algorithm, and its performance degrades much less than SalLoc's when the budget reduces from 30% to 20%; meanwhile, its test accuracy is comparable to other methods (better for the pairing with Reset + Finetune)**.

As discussed above, we choose to determine criticality in a coarser-grained way compared to individual parameters. To that end, for an output "channel" o_i (or "neuron" more broadly, encompassing non-convolutional architectures), we describe how to obtain its criticality score c_{o_i} based on the criticality score of its constituent parameters. Let $\tilde{s}_i$ be a list of the criticality scores for the parameters belonging to neuron o_i, sorted in descending order. We set the neuron criticality c_{o_i} to be the average of the top h scores of its associated parameters: $c_{o_i} = \frac{1}{h}\Sigma_{j=1}^{h}\tilde{s}_i[j]$.

Finally, having obtained the neuron criticality scores, we put together the mask m_α for parameter budget α, represented as a binary vector of size p, where a 1 indicates the corresponding parameter will be updated by the unlearning algorithm, whereas an entry of 0 indicates it will be kept unchanged. To this end, we form another sorted list that sorts the neurons in descending order of their criticality scores. We then pick the largest number of neurons from the start of the sorted list, such that the total selected parameters are within the budget. Then, we assign a 1 to all entries of m_α for parameters belonging to the chosen critical neurons and 0 to the rest. We provide pseudocode in Section A.2.

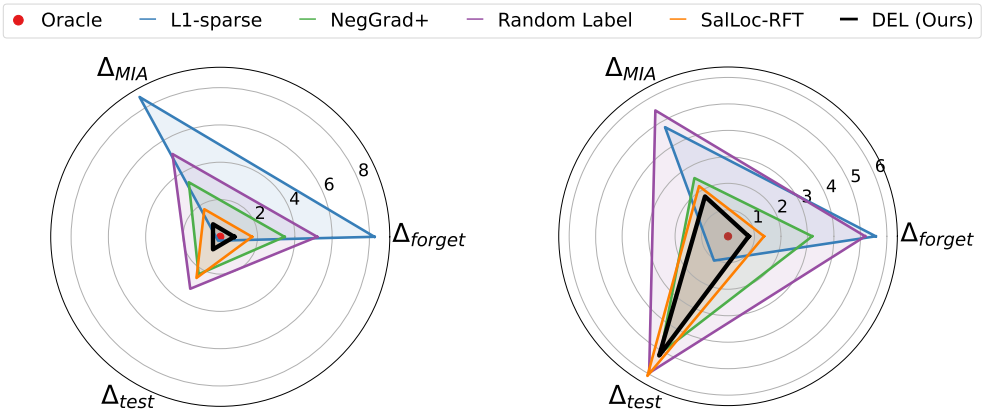
Our localization strategy can, in principle, be paired with any unlearning algorithm, and it in fact pairs well with several of them (see Section 6). We obtain the best results by pairing it with the simple Reset + Finetune (RFT) algorithm. We refer to the combination of our localization strategy with RFT as **Deletion by Example Localization (DEL)**.

6 Comparison to state-of-the-art and analyses

We carry out comprehensive experiments on three datasets and architectures (CIFAR-10 with ResNet-18, SVHN with ViT and ImageNet-100 with ResNet-50; see Section A.3 for details), to examine the performance of our localization strategy paired with various unlearning algorithms, on different forget sets. We also conduct analyses on the factors determining the success of localized unlearning.

Our strategy outperforms the rest paired with *any* unlearning algorithm considered. From Figure 3, we observe that i) for each considered unlearning algorithm, our method yields the best results on the unlearning metrics (forget accuracy and MIA score), ii) at the same time, its test accuracy is comparable to others in general, and better than others when paired with the Reset + Finetune unlearning algorithm.

DEL is the most robust to the parameter budget. Figure 2 shows that DEL most consistently yields strong results across several budgets, compared to other strategies. Figure 3 corroborates this finding in



(a) SVHN/ViT (b) ImageNet-100/ResNet-50

Figure 4: **On SVHN / ViT and ImageNet100 / ResNet-50, DEL outperforms full-parameter and localized methods in terms of unlearning quality**. For each of the three axes(Δ_{forget}, Δ_{test}, Δ_{MIA}), values closer to the center represent better performance as they are closer to the Oracle (red point at the center). L1-sparse has better test accuracy than DEL but has poor unlearning performance. This is for non-IID and IID forget sets on SVHN / ViT and ImageNet100 respectively, and $\alpha = 30\%$ for localized methods; see Tables 12 and 13 for full results.

the context of different unlearning algorithms too, showing that our localization method experiences much lower performance degradation compared to SalLoc, when the budget is reduced: we significantly outperform SalLoc when the budget is 20%.

Table 2: **Subclass unlearning** on CIFAR-10 / ResNet-18. DEL outperforms the state-of-the-art localized and full-parameter unlearning on CIFAR-10 / ResNet-18 on two forget sets in the subclass unlearning setting. For each metric m, $\Delta_m = m_{\text{oracle}} - m_{\text{unlearn}}$; lower absolute value is better. Note that SalLoc-RL corresponds to "SalUn" which uses Random Label, regarded as state-of-the-art.

		Non-IID Forget Set			IID Forget Set		
		$\Delta_{\text{forget}} \downarrow$	$\Delta_{\text{MIA}} \downarrow$	$\Delta_{\text{test}} \downarrow$	$\Delta_{\text{forget}} \downarrow$	$\Delta_{\text{MIA}} \downarrow$	$\Delta_{\text{test}} \downarrow$
Full-parameter Unlearning	Retraining (Oracle)	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
	Fine-tuning	$-5.60_{\pm 0.89}$	$6.07_{\pm 1.10}$	$\mathbf{-1.61_{\pm 0.34}}$	$-1.98_{\pm 1.10}$	$1.96_{\pm 1.11}$	$-2.00_{\pm 0.72}$
	NegGrad+	$-4.44_{\pm 0.95}$	$4.92_{\pm 1.14}$	$4.61_{\pm 0.22}$	$1.87_{\pm 1.31}$	$1.89_{\pm 1.30}$	$4.21_{\pm 0.67}$
	NegGrad	$-3.30_{\pm 0.72}$	$3.76_{\pm 0.94}$	$4.60_{\pm 0.18}$	$-15.04_{\pm 0.40}$	$15.39_{\pm 0.39}$	$-1.11_{\pm 0.51}$
	Random Label	$-1.64_{\pm 0.98}$	$2.07_{\pm 1.15}$	$4.33_{\pm 0.19}$	$1.69_{\pm 0.46}$	$1.69_{\pm 0.47}$	$4.93_{\pm 0.47}$
	L1-sparse	$-1.50_{\pm 0.82}$	$-1.01_{\pm 1.03}$	$2.07_{\pm 0.51}$	$1.80_{\pm 1.20}$	$-1.80_{\pm 1.08}$	$\mathbf{0.62_{\pm 0.67}}$
	IU[4]	$-5.00_{\pm 0.88}$	$5.04_{\pm 0.91}$	$4.18_{\pm 0.19}$	$-2.20_{\pm 0.39}$	$2.19_{\pm 0.38}$	$10.94_{\pm 0.43}$
Localized Unlearning	SSD	$-11.16_{\pm 6.28}$	$11.18_{\pm 6.29}$	$2.68_{\pm 1.18}$	$1.60_{\pm 1.99}$	$1.59_{\pm 1.98}$	$11.58_{\pm 1.03}$
	CritMem-RL ($\alpha = 16\%$)	$-1.82_{\pm 1.19}$	$1.87_{\pm 1.21}$	$4.86_{\pm 0.19}$	$-2.03_{\pm 0.45}$	$2.05_{\pm 0.45}$	$4.36_{\pm 0.37}$
	Shallowest-RL ($\alpha = 25\%$)	$1.29_{\pm 1.62}$	$-0.80_{\pm 1.74}$	$5.88_{\pm 0.18}$	$3.41_{\pm 0.77}$	$-3.43_{\pm 0.78}$	$6.43_{\pm 0.52}$
	SalLoc-RL ($\alpha = 30\%$)	$-2.80_{\pm 1.45}$	$3.30_{\pm 1.54}$	$4.63_{\pm 0.27}$	$-3.81_{\pm 0.40}$	$3.80_{\pm 0.39}$	$4.29_{\pm 0.45}$
	DEL ($\alpha = 30\%$)	$0.43_{\pm 1.06}$	$0.64_{\pm 1.23}$	$2.23_{\pm 0.25}$	$0.97_{\pm 0.42}$	$-0.97_{\pm 0.40}$	$1.87_{\pm 0.49}$

DEL performs strongly across datasets / architectures, unlearning settings, and forget set types. We compare DEL to state-of-the-art methods for unlearning, including ones that update all parameters, on different datasets. We also consider three different unlearning settings: subclass unlearning, class unlearning, and continual unlearning, as well as two forget set types in subclass unlearning setting: IID and non-IID forget

[4]The code is adapted from https://github.com/OPTML-Group/Unlearn-Saliency/

Table 3: **Class Unlearning** on CIFAR-10 / ResNet-18. DEL outperforms the state-of-the-art methods in class unlearning setting (specifically for classes 2 and 5) on CIFAR-10 / ResNet-18.

	Class Unlearning (class 2)		**Class Unlearning (class 5)**	
Method	$\Delta_{\textbf{forget}} \downarrow$	$\Delta_{\textbf{test}} \downarrow$	$\Delta_{\textbf{forget}} \downarrow$	$\Delta_{\textbf{test}} \downarrow$
Retraining (Oracle)	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
Fine-tuning	$-0.05_{\pm 0.04}$	$-1.63_{\pm 0.21}$	$-3.78_{\pm 1.19}$	$-1.46_{\pm 0.13}$
NegGrad+	$-3.70_{\pm 0.13}$	$0.75_{\pm 0.18}$	$-6.92_{\pm 0.13}$	$1.95_{\pm 0.22}$
NegGrad	$-3.30_{\pm 0.13}$	$0.92_{\pm 0.25}$	$-6.73_{\pm 0.06}$	$2.09_{\pm 0.09}$
Random Label	$-11.25_{\pm 0.44}$	$\mathbf{0.64_{\pm 0.21}}$	$-11.93_{\pm 1.39}$	$\mathbf{0.04_{\pm 0.14}}$
L1-sparse	$\mathbf{0.00_{\pm 0.00}}$	$-1.78_{\pm 0.23}$	$\mathbf{0.00_{\pm 0.00}}$	$-1.86_{\pm 0.23}$
Shallowest-RL ($\alpha = 25\%$)	$4.02_{\pm 0.62}$	$6.63_{\pm 0.39}$	$-2.27_{\pm 0.10}$	$6.59_{\pm 0.57}$
SalLoc-RL ($\alpha = 30\%$)	$-12.72_{\pm 0.12}$	$2.07_{\pm 0.20}$	$-10.97_{\pm 0.01}$	$1.17_{\pm 0.12}$
DEL ($\alpha = 30\%$)	$\mathbf{0.00_{\pm 0.00}}$	$0.79_{\pm 0.23}$	$\mathbf{0.00_{\pm 0.00}}$	$0.76_{\pm 0.25}$

Table 4: **Continual Unlearning**, on CIFAR-10 / ResNet-18. DEL outperforms the state-of-the-art methods in a continual unlearning setting (specifically, unlearning classes 2 followed by class 5) on CIFAR-10 / ResNet-18.

	Retraining (Oracle)	Fine-tuning	L1-sparse	Shallowest-RFT ($\alpha = 25\%$)	SalLoc-RFT ($\alpha = 30\%$)	**DEL** ($\alpha = 30\%$)
$\Delta_{\text{forget}} \downarrow$	$0.00_{\pm 0.00}$	$-9.99_{\pm 0.57}$	$-3.66_{\pm 0.41}$	$-4.39_{\pm 0.76}$	$3.80_{\pm 1.53}$	$\mathbf{-0.83_{\pm 0.49}}$
$\Delta_{\text{test}} \downarrow$	$0.00_{\pm 0.00}$	$-1.73_{\pm 0.19}$	$-2.87_{\pm 0.27}$	$5.71_{\pm 0.71}$	$4.03_{\pm 1.41}$	$\mathbf{0.76_{\pm 0.16}}$
$\Delta_{\text{MIA}} \downarrow$	$0.00_{\pm 0.00}$	$9.96_{\pm 0.59}$	$3.67_{\pm 0.37}$	$4.40_{\pm 0.75}$	$-3.83_{\pm 1.52}$	$\mathbf{0.81_{\pm 0.45}}$

sets (details in Section A.3). An **IID forget set** comprised of 10% of randomly-chosen training samples, and a **non-IID forget set** of the same size, but choosing samples belonging to a subset of the classes. We present the results in Table 2, Table 3, Table 4, Figure 4 as well as in Table 12, Table 13 and Table 18. For each localized unlearning strategy and on each dataset, we report results using its best identified parameter budget and its best-paired unlearning algorithm for that setting. We find that DEL is the most consistent method in delivering strong results across the board. On all settings considered in CIFAR-10 / ResNet-18 and ImageNet100/ResNet-50, it sets a new state-of-the-art in terms of both unlearning metrics, while also outperforming all localized methods in terms of test accuracy too. The same is true for SVHN's Non-IID forget set, while on the IID forget set, we observe that exceptionally, some full-parameter methods outperform DEL (and other localized methods) in one unlearning metric (but not the other). We later investigate why localized unlearning yields less pronounced gains on IID forget sets. Overall, DEL is the most consistent approach in achieving the strongest unlearning results, with the smallest test accuracy drop compared to localized methods.

Furthermore, we investigate the scalability of DEL by introducing two additional forget set sizes, comprising 30% and 60% of randomly selected samples from the training set of the CIFAR-10/ResNet-18 experiment. As presented in Table 16 in the Appendix, DEL consistently demonstrates robustness across varying forget set sizes and achieves higher performance than the competitors in all settings.

Localized unlearning succeeds due to selecting critical parameters, not just a smaller number of them. We design a control experiment to investigate to what extent the success of localized unlearning depends on *which* parameters are chosen (rather than simply *how many*). To this end, we compare the mask produced by each localization strategy to a "random mask" that is constructed to follow the same structure and distribution of the number of chosen parameters per layer as the corresponding non-random mask. For example, to create the random mask that CritMem will be compared with, we randomly select a number of

channels for each layer equal (but randomly selected this time) to the number of channels that CritMem selects for that layer. Table 5 indicates that success in unlearning metrics is indeed due to pinpointing critical parameters but interestingly, a random selection of parameters is better for the test accuracy metric. This is likely due to the fact that parameters that are critical for the forget set are generally important (for all examples), so resetting them causes a drop in test accuracy too. This highlights a fundamental tension in unlearning (unlearning some examples while preserving performance).

Table 5: **Control experiment: comparison to a random mask** on CIFAR-10 / ResNet-18 (RFT unlearning, $\alpha = 16\%$). Lower absolute values of Δ indicate better performance.

	Standard Masking			**Random Masking**		
	CritMem	**SalLoc**	**Ours**	**CritMem**	**SalLoc**	**Ours**
$\Delta_{\text{forget}} \downarrow$	$-13.25_{\pm 1.53}$	$-8.26_{\pm 0.92}$	$-2.86_{\pm 1.09}$	$-18.60_{\pm 1.03}$	$-13.26_{\pm 0.80}$	$-8.58_{\pm 0.98}$
$\Delta_{\text{MIA}} \downarrow$	$13.73_{\pm 1.62}$	$8.75_{\pm 1.09}$	$3.36_{\pm 1.27}$	$19.09_{\pm 1.21}$	$13.78_{\pm 1.02}$	$9.07_{\pm 1.17}$
$\Delta_{\text{test}} \downarrow$	$2.85_{\pm 0.31}$	$3.50_{\pm 0.53}$	$2.62_{\pm 0.22}$	$1.99_{\pm 0.19}$	$1.91_{\pm 0.57}$	$1.89_{\pm 0.21}$

Is localized unlearning better due to tailoring to $\mathcal{S}$? We design a set of experiments to investigate this by changing the criticality criterion. Specifically, we choose two criteria that are not specific to the forget set $\mathcal{S}$: the first uses only the magnitude of the weights ("weights"), and the second uses weighted gradients, but where the gradients this time are over all of $\mathcal{D}_{\text{train}}$ rather than just $\mathcal{S}$ ("Weighted gradients (train set)"). Our rationale is that, if either of these forget-set-agnostic criteria works equally well as our method's criterion ("Weighted gradients (forget set)"), this would suggest that the success of our method is not due to specialization to $\mathcal{S}$ but rather finding parameters that are "generally critical" for the training data. We observe from Table 6 that, for the IID forget set, the above two criteria that depend on $\mathcal{D}_{\text{train}}$ rather than $\mathcal{S}$ specifically, yield more similar results to our criterion. This is reasonable since the forget and train follow the same distribution in the IID forget set case. On the other hand, for the non-IID forget set, we do observe that tailoring the criticality criterion specifically to $\mathcal{S}$ yields better unlearning performance. We take away that i) the success of different localization strategies is dependent on the distribution of the forget set, ii) our method (shaded gray area in the table) is a top performer in all cases.

7 Discussion and Conclusion

We investigated localized unlearning informed by hypotheses for where memorization happens, leading to a new localization strategy that is more efficient than the algorithm from the memorization literature that it builds upon, while outperforming prior work on several metrics, when paired with different unlearning algorithms. DEL (the pairing our localization strategy with reset-and-finetune unlearning), is a top performer on different forget sets, datasets and architectures and parameter budgets. We find that the success of localized unlearning in terms of forgetting quality is indeed due to pinpointing critical parameters to act on, rather than a smaller number. But our results also suggest that parameters that are critical for the forget set may be critical overall, leading to fundamental tensions between unlearning while preserving utility. We also find that for non-IID forget sets, tailoring the parameter selection to the specific forget set (rather than the training set more broadly) is more important than it is for IID forget sets. DEL is a top performer in all scenarios.

So, does locating memorization inform unlearning? Hase et al. (2024) find that, for model editing in LLMs, the "causal tracing" method (Meng et al., 2022) for knowledge localization, surprisingly, does not indicate which layer to modify in order to most successfully rewrite a stored fact with a new one. That is, they find that success in editing tasks is generally unrelated to localization results based on causal tracing. Guo et al. (2024) study whether mechanistic interpretability insights improve unlearning of "factual associations" in LLMs. They also find that localization techniques based on preserving outputs (such as causal tracing) yield performance that is no better, or even worse, than non-localized unlearning. However, they come up with a mechanistic unlearning method that does outperform both output-based localization and non-localized unlearning, showing that some form of localization is useful. Our results, in the very different context of

Table 6: Investigation of different granularity and criticality criteria in a non-iterative localization algorithm (α=15%) on **IID** and **Non-IID forget set**. The shaded region corresponds to our method.

Granularity		Non-IID forget set				IID forget set			
		Gradients (forget set)	Weights	Weighted gradients (train set)	Weighted gradients (forget set)	Gradients (forget set)	Weights	Weighted gradients (train set)	Weighted gradients (forget set)
Individual parameter	$\Delta_{\text{forget}} \downarrow$	$\mathbf{-7.58_{\pm 8e-3}}$	$15.50_{\pm 6e-3}$	$14.03_{\pm 0.01}$	$15.07_{\pm 0.01}$	$\mathbf{-6.72_{\pm 0.41}}$	$11.67_{\pm 0.51}$	$11.19_{\pm 0.15}$	$11.15_{\pm 0.48}$
	$\Delta_{\text{test}} \downarrow$	$\mathbf{3.63_{\pm 4e-3}}$	$10.80_{\pm 5e-3}$	$10.60_{\pm 6e-3}$	$11.55_{\pm 6e-3}$	$\mathbf{1.98_{\pm 0.45}}$	$11.11_{\pm 0.60}$	$10.90_{\pm 0.51}$	$11.05_{\pm 0.54}$
	$\Delta_{\text{MIA}} \downarrow$	$\mathbf{8.74_{\pm 1.10}}$	$-14.73_{\pm 0.97}$	$-13.89_{\pm 1.29}$	$-16.32_{\pm 1.39}$	$\mathbf{6.81_{\pm 0.41}}$	$11.68_{\pm 0.50}$	$-11.21_{\pm 0.52}$	$-11.13_{\pm 0.46}$
Output channel	$\Delta_{\text{forget}} \downarrow$	$-12.33_{\pm 8e-3}$	$6.02_{\pm 5e-3}$	$-2.52_{\pm 0.01}$	$\mathbf{1.58_{\pm 0.01}}$	$-5.18_{\pm 0.40}$	$1.33_{\pm 0.41}$	$1.69_{\pm 0.68}$	$\mathbf{0.65_{\pm 0.60}}$
	$\Delta_{\text{test}} \downarrow$	$\mathbf{1.55_{\pm 2e-3}}$	$5.64_{\pm 2e-3}$	$3.52_{\pm 4e-3}$	$4.41_{\pm 2e-3}$	$\mathbf{1.26_{\pm 0.54}}$	$2.86_{\pm 0.45}$	$2.27_{\pm 0.44}$	$3.05_{\pm 0.43}$
	$\Delta_{\text{MIA}} \downarrow$	$15.53_{\pm 1.10}$	$\mathbf{-1.02_{\pm 0.93}}$	$6.73_{\pm 1.12}$	$3.33_{\pm 1.28}$	$5.16_{\pm 0.39}$	$-1.40_{\pm 0.39}$	$1.67_{\pm 0.68}$	$\mathbf{0.69_{\pm 0.59}}$

unlearning a subset of data in vision classifiers, offer an important data point in this ongoing discussion. In line with Hase et al. (2024), we find that directly translating memorization hypotheses into localization strategies does not help unlearning: Deepest led to very poor unlearning results (demonstrating either its weakness as a memorization locator, or the disconnect between memorization localization and unlearning performance), and CritMem, while showing more promise, did not perform better than simple baselines, while being significantly more expensive than them. However, insights from Maini et al. (2023), in particular regarding the granularity and criticality criterion used during localization led us to improve upon the state-of-the-art localized and full-parameter unlearning methods, renewing hopes that, while memorization localization and unlearning may be separate research questions, progress in the former may guide progress in the latter.

Acknowledgments

This work was supported by a Google Ph.D. Fellowship to Reihaneh Torkzadehmahani. The authors of this work take full responsibility for its content

References

Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. In *International conference on machine learning*, pp. 233–242. PMLR, 2017.

Idan Attias, Gintare Karolina Dziugaite, Mahdi Haghifam, Roi Livni, and Daniel M Roy. Information complexity of stochastic convex optimization: Applications to generalization and memorization. *arXiv preprint arXiv:2402.09327*, 2024.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Robert Baldock, Hartmut Maennel, and Behnam Neyshabur. Deep learning through the lens of example difficulty. *Advances in Neural Information Processing Systems*, 34:10876–10889, 2021.

Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*, pp. 141–159. IEEE, 2021.

Gavin Brown, Mark Bun, Vitaly Feldman, Adam Smith, and Kunal Talwar. When is memorization of irrelevant training data necessary for high-accuracy learning? In *Proceedings of the 53rd annual ACM SIGACT symposium on theory of computing*, pp. 123–132, 2021.

Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *2015 IEEE symposium on security and privacy*, pp. 463–480. IEEE, 2015.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Cynthia Dwork. Differential privacy. In *International colloquium on automata, languages, and programming*, pp. 1–12. Springer, 2006.

Chongyu Fan, Jiancheng Liu, Yihua Zhang, Dennis Wei, Eric Wong, and Sijia Liu. Salun: Empowering machine unlearning via gradient-based weight saliency in both image classification and generation. *arXiv preprint arXiv:2310.12508*, 2023.

Vitaly Feldman. Does learning require memorization? a short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pp. 954–959, 2020.

Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. *Advances in Neural Information Processing Systems*, 33:2881–2891, 2020.

Jack Foster, Stefan Schoepf, and Alexandra Brintrup. Fast machine unlearning without retraining through selective synaptic dampening. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 12043–12051, 2024.

Antonio Ginart, Melody Guan, Gregory Valiant, and James Y Zou. Making ai forget you: Data deletion in machine learning. *Advances in neural information processing systems*, 32, 2019.

Shashwat Goel, Ameya Prabhu, Amartya Sanyal, Ser-Nam Lim, Philip Torr, and Ponnurangam Kumaraguru. Towards adversarial evaluations for inexact machine unlearning. *arXiv preprint arXiv:2201.06640*, 2022.

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*, pp. 9304–9312, 2020a.

Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Forgetting outside the box: Scrubbing deep networks of information accessible from input-output observations. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIX 16*, pp. 383–398. Springer, 2020b.

Laura Graves, Vineel Nagisetty, and Vijay Ganesh. Amnesiac machine learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 11516–11524, 2021.

Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. Certified data removal from machine learning models. *arXiv preprint arXiv:1911.03030*, 2019.

Phillip Huang Guo, Aaquib Syed, Abhay Sheshadri, Aidan Ewart, and Gintare Karolina Dziugaite. Robust knowledge unlearning via mechanistic localizations. In *ICML 2024 Next Generation of AI Safety Workshop*, 2024.

Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. *Advances in Neural Information Processing Systems*, 36, 2024.

Jamie Hayes, Ilia Shumailov, Eleni Triantafillou, Amr Khalifa, and Nicolas Papernot. Inexact unlearning needs more careful evaluations to avoid a false sense of privacy. *arXiv preprint arXiv:2403.01218*, 2024.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

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*, pp. 2008–2016. PMLR, 2021.

Matthew Jagielski, Om Thakkar, Florian Tramer, Daphne Ippolito, Katherine Lee, Nicholas Carlini, Eric Wallace, Shuang Song, Abhradeep Thakurta, Nicolas Papernot, et al. Measuring forgetting of memorized training examples. *arXiv preprint arXiv:2207.00099*, 2022.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pp. 1885–1894. PMLR, 2017.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Meghdad Kurmanji, Peter Triantafillou, Jamie Hayes, and Eleni Triantafillou. Towards unbounded machine unlearning. *Advances in neural information processing systems*, 36, 2024.

Jiancheng Liu, Parikshit Ram, Yuguang Yao, Gaowen Liu, Yang Liu, PRANAY SHARMA, Sijia Liu, et al. Model sparsity can simplify machine unlearning. *Advances in Neural Information Processing Systems*, 36, 2024.

Pratyush Maini, Michael C Mozer, Hanie Sedghi, Zachary C Lipton, J Zico Kolter, and Chiyuan Zhang. Can neural network memorization be localized? In *International Conference on Machine Learning*, 2023.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372, 2022.

Seth Neel, Aaron Roth, and Saeed Sharifi-Malvajerdi. Descent-to-delete: Gradient-based methods for machine unlearning. In *Algorithmic Learning Theory*, pp. 931–962. PMLR, 2021.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, Andrew Y Ng, et al. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, pp. 4. Granada, 2011.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 33:19920–19930, 2020.

Seonguk Seo, Dongwan Kim, and Bohyung Han. Revisiting machine unlearning with dimensional alignment. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 3206–3215. IEEE, 2025.

Cory Stephenson, Suchismita Padhy, Abhinav Ganesh, Yue Hui, Hanlin Tang, and SueYeon Chung. On the geometry of generalization and memorization in deep neural networks. *arXiv preprint arXiv:2105.14602*, 2021.

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)*, pp. 303–319. IEEE, 2022.

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. An empirical study of example forgetting during deep neural network learning. *arXiv preprint arXiv:1812.05159*, 2018.

Eleni Triantafillou, Peter Kairouz, Fabian Pedregosa, Jamie Hayes, Meghdad Kurmanji, Kairan Zhao, Vincent Dumoulin, Julio Jacques Junior, Ioannis Mitliagkas, Jun Wan, et al. Are we making progress in unlearning? findings from the first neurips unlearning competition. *arXiv preprint arXiv:2406.09073*, 2024.

Phil Wang. Vit for small datasets. https://github.com/lucidrains/vit-pytorch/blob/main/vit_pytorch/vit_for_small_dataset.py, 2021.

Haonan Yan, Xiaoguang Li, Ziyao Guo, Hui Li, Fenghua Li, and Xiaodong Lin. Arcane: An efficient architecture for exact machine unlearning. In *IJCAI*, volume 6, pp. 19, 2022.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

Kairan Zhao, Meghdad Kurmanji, George-Octavian Bărbulescu, Eleni Triantafillou, and Peter Triantafillou. What makes unlearning hard and what to do about it. *arXiv preprint arXiv:2406.01257*, 2024.

A Appendix

A.1 Unlearning definition

In this section, we discuss an alternative formal definition of unlearning, proposed in Ginart et al. (2019); Neel et al. (2021), using a notion closely related to Differential Privacy (Dwork, 2006).

Definition A.1. Unlearning-2. For a training algorithm $\mathcal{A}$, an algorithm $\mathcal{U}$ is an (ϵ, δ)-*unlearner* if, for any training dataset $\mathcal{D}_{\text{train}}$ and forget set $\mathcal{S}$, the distributions of $\mathcal{A}(\mathcal{D}_{\text{train}} \setminus \mathcal{S})$ and $\mathcal{U}(\theta^o, \mathcal{S}, \mathcal{D}_{\text{train}} \setminus \mathcal{S})$ are (ϵ, δ)-close, where we say two distributions μ, ν are (ϵ, δ)-*close* if $\mu(B) \leq e^\epsilon \nu(B) + \delta$ and $\nu(B) \leq e^\epsilon \mu(B) + \delta$ for all events B.

Intuitively, the above compares (the distribution of) models that are obtained by two different recipes to one another:

- $\mathcal{A}(\mathcal{D}_{\text{train}} \setminus \mathcal{S})$, retraining "from scratch" on only the retain set, which is prohibitively expensive but ideal from the standpoint of eliminating the influence of $\mathcal{S}$ on the model, and
- $\mathcal{U}(\theta^o, \mathcal{S}, \mathcal{D}_{\text{train}} \setminus \mathcal{S})$, applying $\mathcal{U}$ to post-process the original model θ^o in order to unlearn $\mathcal{S}$.

A successful unlearning algorithm $\mathcal{U}$ is one that causes these two recipes to yield similar models, with the second recipe being substantially more computationally-efficient compared to the first, in order to justify paying the cost of approximate unlearning rather than simply using the first recipe directly.

Note that we refer to *distributions* here since re-running either of the two recipes with a different random seed, that controls the initialization or the order of mini-batches, for example, would yield slightly different model weights in each case. The above definition therefore measures unlearning quality based on the notion of (ϵ, δ)-*closeness* between the two distributions. The smaller ϵ and δ are, indicating increased closeness, the better the unlearning algorithm.

Relationship and differences to our definition This definition compares distributions in weight space, whereas our Definition 2.1 compares distributions of *outputs* of models on the forget set. We opted for the latter in the main paper as it more closely reflects the metrics we use for evaluation (which are the standard metrics used in unlearning papers). Note that, even works that adopt definitions in weight-space end up operationalizing them using outputs of models (Triantafillou et al., 2024) instead of performing weight-space comparisons. This is for several reasons: comparing weights of models directly may be inappropriate since neural networks are permutation-invariant. Weight space is also much higher dimensional, posing challenges in creating the right metrics, and, finally, ultimately what we may care about for various applications of interest is the "behaviours" (e.g. predictions) of models, rather than their weights. Definition 2.1 captures this more directly.

A.2 Pseudocode

A.3 Experimental Setup

Datasets The CIFAR-10 dataset (Krizhevsky et al., 2009) consists of $50,000$ train and $10,000$ test images of shape 32×32 from 10 classes. The SVHN dataset (Netzer et al., 2011) includes $73,257$ train and $26,032$

Algorithm 1: Our localization strategy

Input: Original model θ^o with p parameters, forget set $\mathcal{S}$, parameter budget α
Output: Localization mask m_α

```
// Compute criticality score for each parameter
```
for $j \leftarrow 1$ **to** p **do**
  $s_j \leftarrow 0$;
for *Mini-batch* $\mathcal{B} \in \mathcal{S}$ **do**
  for $j \leftarrow 1$ **to** p **do**
    $s_j \leftarrow s_j + \theta^o_j \cdot g_j(\theta^o, \mathcal{B})$;

```
// Consider only the magnitude of each weighted gradient
```
for $j \leftarrow 1$ **to** p **do**
  $s_j \leftarrow |s_j|$;

```
// Compute criticality score for each output channel/neuron
```
$\tilde{s} \leftarrow \text{Sort}(s)$;
for $o_i \in \theta^o$ **do**
  $c_{o_i} \leftarrow \frac{1}{h} \sum_{j=1}^{h} \tilde{s}_i[j]$;

```
// Construct localization mask
```
$\tilde{c} \leftarrow \text{Sort}(c)$;
$m_\alpha \leftarrow \mathbf{1}\left(\sum_{i=1}^{j} |\tilde{c}_{o_i}| \leq (p \cdot \alpha)\right)$;

test samples. The samples are of shape 32×32 pixel, and from 10 classes. The ImageNet-100 (Hugging Face version) dataset is a subset of ImageNet (Deng et al., 2009), containing 126,689 train and 5,000 test samples from 100 classes, randomly selected from the original ImageNet classes. The resolution of the images on the shortest side is 160 pixels.

We perform no preprocessing or augmentation on the images of CIFAR-10 and SVHN, except dividing the feature values by 255. For ImageNet-100, on the other hand, we randomly crop the train images to size 128×128, and horizontally flip them. Moreover, we first resize the test images to 160×160, and then center-crop them to 128×128. We normalize the features of both train and test images with the mean and variance of ImageNet.

For the unlearning experiments, we employ three different unlearning settings: (1) sublcalss unlearning, (2) class unlearning, and (3) continual unlearning. For subclass unlearning, we consider two different forget sets: (1) IID, where we uniformly select approximately 10 percent of the images from the train set, and (2) NonIID, in which we randomly select half of the samples from two classes (2 and 5 in CIFAR-10, and 3 and 6 in SVHN) so that the size of the forget set is almost 10 percent of the train set. Note that in the former, the distributions of the samples in the forget and retain sets are highly similar, whereas in the latter, the sample distribution of the forget set is very different from that of the retain set. In class unlearning, the forget set comprises all training samples from the target class (e.g., class 2 and class 5 in CIFAR-10). However, continual unlearning involves sequentially adding a randomly selected half of the samples from each target class to the forget set. More specifically, for the first unlearning task—unlearning class 2 from the pretrained model—the forget set consists of a randomly selected half of the samples from class 2 in CIFAR-10. For the second unlearning task—unlearning classes 2 and 5 from the model, already unlearned class 2—the forget set includes a randomly selected half of the samples from both classes 2 and 5 in CIFAR-10. We further extended our continual unlearning experiments on the CIFAR-10 / ResNet-18 setting by adopting a longer forgetting sequence (class 2 → class 5 → class 3 → class 4).

Models We capitalize on the original implementation of ResNet-18 and ResNet-50 (He et al., 2016) from PyTorch and the implementation of Vision Transformer (ViT) (Dosovitskiy, 2020) from Wang (2021). ResNet-18 and ViT contain around 11 million parameters, whereas ResNet-50 has approximately 25 million parameters.

Due to the low-resolution nature of CIFAR-10, we replace the first convolutional layer of ResNet-18 with a new convolutional layer with kernel size of 3×3, and remove the max-pooling layer.

Note that the architectures of the considered models are very different from each other. ResNet-18/50 are convolutional (Conv) networks with an input Conv layer, multiple residual blocks, and a final classifier layer. The normalization layer of ResNet-18/50 is batch normalization (Ioffe & Szegedy, 2015). In ResNet-18/50, the input images are downsampled multiple times so that deeper layers operate on smaller input tensors. However, deeper layers have more filters (and thus, more trainable parameters) than shallower layers.

The ViT architecture, on the other hand, employs linear (fully-connected) and multi-head attention layers as its main building blocks. It first divides the input images into square patches (e.g., of shape 8×8) and gives them as tokens (after some preprocessing, including positional encoding) to the encoder blocks. No downsampling is performed on the input tensors by the encoder blocks. Moreover, all (i.e., both deeper and shallower) encoder blocks have an identical number of trainable parameters. The normalization layer of ViT architectures is layer normalization (Ba et al., 2016).

In more detail, our ViT model employs patch size of 8×8 and contains 4 transformer layers with hidden size of 512. Each layer consists of a multi-head attention block and a projection block. The number of heads and the head size in the attention blocks are 8 and 64, respectively. The multilayer perceptron (MLP) size of the project blocks is 1536. The dropout probability of the dropout layers is zero. The CLS token is used for the classification. The model size is around 11 million parameters. Note that we used the terminogly (e.g., hidden size and MLP size) from Table 1 of the original ViT paper (Dosovitskiy, 2020) in our model description.

Training For the original (pretrained) models, we train ResNet-18 on CIFAR-10 and ViT on SVHN (i.e. on the training set of the datasets) for 50 epochs using the SGD optimizer with momentum of 0.9, cross-entropy loss function, and batch size of 128. The base learning rate values are 0.1 and 0.05 for CIFAR-10 / ResNet-18 and SVHN / ViT, respectively, which are gradually decayed by a factor of 0.01 using the Cosine Annealing scheduler. For the oracle model (gold standard), we train the model from scratch only on the retain set, following the same procedure employed for the pretrained model, except the number of epochs, which we set to 20, and the learning rate, which is half of that in the original training. In the tables below, we provide the hyper-parameter values for the approximate unlearning algorithms. We repeat each experiment three times and report the average values along with 95% confidence interval.

Table 7: Learning rate tuning.

	Scheduler	Parameters
Finetuning/ l1-sparse	CosineAnnealingLR	$\eta_{min} = 0.01 * lr_{init}$
Random Label	CosineAnnealingLR	$\eta_{min} = 0.5 * lr_{init}$
NegGrad+/NegGrad	Constant	-

	Non-IID forget set		IID Forget set	
	lr_{best}	candidate values	lr_{best}	Candidate values
Finetuning	1.25	[0.5, 1.5]	1.25	[0.5, 1.5]
l1-sparse	0.5	[0.1, 1]	0.5	[0.1, 1]
Random Label	7e-3	[5e-3, 1e-2]	6e-3	[5e-3, 1e-2]
NegGrad+	7e-4	[5e-4, 1e-3]	0.14	[0.1, 1]
NegGrad	7e-6	[5e-6, 1e-5]	4e-3	[1e-3, 5e-3]
CritMem-RL ($\alpha = 16\%$)	0.02	[0.01, 0.1]	0.02	[0.01, 0.1]
Shallowest-RL ($\alpha = 25\%$)	7e-3	[5e-3, 1e-2]	7e-3	[5e-3, 1e-2]
SalLoc-RL ($\alpha = 30\%$)	0.012	[5e-3, 1e-2]	0.012	[5e-3, 1e-2]
DEL ($\alpha = 30\%$)	0.015	[5e-3, 1e-2]	0.015	[5e-3, 1e-2]

The SSD algorithm incorporates two key hyperparameters: α, which governs parameter selection, and λ, which controls weight dampening. Table 8 summarizes the best hyper-parameter tuned for each setting, along with the resulting parameter budgets.

Table 8: Hyper-parameter tuning of the SSD algorithm and the corresponding parameter budgets.

	Non-IID forget set			IID Forget set		
dataset/model	α_{best}	λ_{best}	parameter budget	α_{best}	λ_{best}	parameter budget
CIFAR-10/ResNet-18	16	1	0.5%	1	1	28%
SVHN/ViT	5	3	0.1%	5	3	33%
ImageNet-100/ResNet-50	-	-	-	2	1	2%

A.4 Metrics

Following Fan et al. (2023), we employ accuracy and membership inference attack (MIA) score to evaluate the effectiveness of different unlearning algorithms.

To compute MIA score, a support vector classifier (SVC) is trained on top of outputs coming from the unlearned model for the task of predicting whether an example was used in training or not. This is performed through supervised learning where the test set is used as "unseen data" and a subset of retain set (with the same size as and similar label distribution to the test set) as "seen data". Specifically, the SVC is trained on the "prediction" outputs (i.e. the integers representing the index of the predicted class), aiming to distinguish predictions from seen versus unseen data. Then, the trained classifier is utilized to predict if each sample in the forget set belongs to the seen or unseen data on the unlearned model. Given that, MIA score is computed as follows:

$$\text{MIA}_{\text{score}} = \frac{TN}{|\mathcal{S}|},$$

where TN is the number of true negatives, i.e. forget samples that the classifier recognizes as likely unseen data for the unlearned model, and $|\mathcal{S}|$ is the size of the forget set.

Intuitively, a higher value of $\text{MIA}_{\text{score}}$ means that the unlearned model has been more successful in "fooling" the SVC classifier (i.e. the "membership inference attacker") into thinking that the forget set was not used in training. However, to interpret how high we expect $\text{MIA}_{\text{score}}$ to be for an unlearned model, we must consult the reference point of how high this quantity would be for a model retrained from scratch without the forget set. Note that, even in that case of "ideal unlearning", $\text{MIA}_{\text{score}}$ is not necessarily 100%, and in fact it can be much lower than this. This is because, some examples in the forget set might be so "easy" that, even without ever seeing them, the retrained model can still be equally accurate on those examples as it would have been if they were actually included in training. This would lead to its $\text{MIA}_{\text{score}}$ being lower, since some forget set examples would be classified as "seen" by the SVC. For this reason, in our experiments, we use the reference point as the $\text{MIA}_{\text{score}}$ obtained from retrain-from-scratch as the optimal value for this metric. An ideal unlearning algorithm would, therefore, match that value.

A.5 MIA Evaluation

In this section, we present MIA evaluation results using various MIAs for each model-dataset combination. We compare two MIAs that leverage the model's (i) correctness (Table 10) and (ii) confidence (Table 9). Specifically, we train an SVC to distinguish between the seen (train) and unseen (test) data using either (i) the predictions (i.e. the integers representing the index of the predicted class) of the unlearned model on retain and test examples or (ii) the confidences (i.e. the Softmax values associated with these predicted labels) of the unlearned model on these examples. The results for the first variant (correctness-based MIA) are already presented in Tables 2 and 12, with a summary provided in Table 10 in this section. Here, we expand the MIA evaluations by incorporating the second variant (confidence-based MIA).

According to Table 9 and 10, the absolute value of Δ_{MIA} values are larger when using the confidence-based MIA compared to the correctness-based MIA. Since the model's confidence on retain and test samples provides more information than its predictions on these samples, providing the SVC with confidence values results in a stronger MIA than using the predictions.

Additionally, in the CIFAR-10 / ResNet-18 setting, we observe that our localized unlearning algorithm significantly outperforms the other comparison methods across all evaluated MIAs for both IID and non-IID forget sets. Similarly, in the ViT-SVHN setting with non-IID forget sets, our method outperforms the other methods in both correctness-based and confidence-based MIA evaluations. However, when the forget set is IID in SVHN / ViT setting, for both MIA variants there exists a full-parameter unlearning algorithm (sometimes multiple, depending on the type of MIA) that outperforms all the localized unlearning algorithms, including our method, in terms of MIA evaluation. For example, Fine-tuning yields a more effective confidence-based MIA, while Fine-tuning, Random Label, and L1-sparse demonstrate enhanced performance in correctness-based MIA compared to the localized unlearning algorithms. This is an interesting observation that we hope future work will investigate further.

Table 9: **Confidence**-based MIA evaluation (Δ_{MIA})

		CIFAR-10 / ResNet-18		**SVHN / ViT**	
		IID	**Non-IID**	**IID**	**Non-IID**
Full-parameter Unlearning	Retraining(Oracle)	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
	Fine-tuning	$6.24_{\pm 2.18}$	$6.23_{\pm 1.03}$	$-1.97_{\pm 0.32}$	$13.85_{\pm 0.90}$
	NegGrad+	$3.69_{\pm 1.34}$	$11.27_{\pm 1.19}$	$-19.40_{\pm 11.60}$	$3.75_{\pm 1.82}$
	NegGrad	$28.18_{\pm 0.87}$	$10.44_{\pm 0.83}$	$12.47_{\pm 0.25}$	$29.02_{\pm 1.03}$
	Random Label	$-31.55_{\pm 1.30}$	$-19.96_{\pm 1.46}$	$-15.56_{\pm 2.45}$	$-6.44_{\pm 3.58}$
	L1-sparse	$4.35_{\pm 0.88}$	$10.49_{\pm 2.17}$	$-7.74_{\pm 0.58}$	$7.62_{\pm 0.75}$
Localized Unlearning	CritMem-RL ($\alpha = 16\%, 1\%$)	$-31.56_{\pm 1.04}$	$-14.08_{\pm 2.58}$	$13.12_{\pm 0.24}$	$32.80_{\pm 0.71}$
	Shallowest-RL/RFT ($\alpha = 25\%, 30\%$)	$-25.87_{\pm 0.89}$	$-18.39_{\pm 1.45}$	-16.83 ± 1.05	$-2.5_{\pm 8.54}$
	SalLoc-RL/RFT ($\alpha = 30\%$)	$-24.24_{\pm 0.89}$	$-14.15_{\pm 1.76}$	$-17.77_{\pm 0.42}$	$1.67_{\pm 0.92}$
	DEL ($\alpha = 30\%$)	$-0.59_{\pm 0.90}$	-0.90 ± 1.20	$-5.48_{\pm 0.64}$	$1.12_{\pm 0.76}$

Table 10: **Correctness**-based MIA evaluation (Δ_{MIA})

		CIFAR-10 / ResNet-18		**SVHN / ViT**	
		IID	**Non-IID**	**IID**	**Non-IID**
Full-parameter Unlearning	Retraining(Oracle)	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
	Fine-tuning	$1.96_{\pm 1.11}$	$6.07_{\pm 1.10}$	$-1.05_{\pm 0.42}$	$11.22_{\pm 0.65}$
	NegGrad+	$1.89_{\pm 1.38}$	$4.92_{\pm 1.14}$	$-6.99_{\pm 3.53}$	$3.36_{\pm 1.67}$
	NegGrad	$15.39_{\pm 0.39}$	$3.76_{\pm 0.94}$	$8.00_{\pm 0.22}$	$19.70_{\pm 0.53}$
	Random Label	$1.69_{\pm 0.47}$	$2.07_{\pm 1.15}$	$-2.82_{\pm 0.46}$	$5.13_{\pm 1.81}$
	L1-sparse	$-1.80_{\pm 1.08}$	$-1.01_{\pm 1.03}$	$-2.43_{\pm 0.39}$	$8.66_{\pm 0.31}$
Localized Unlearning	CritMem-RL ($\alpha = 16\%, 1\%$)	$2.05_{\pm 0.45}$	$1.87_{\pm 1.21}$	$8.16_{\pm 0.22}$	$20.71_{\pm 0.26}$
	Shallowest-RL/RFT ($\alpha = 25\%, 30\%$)	$-3.43_{\pm 0.78}$	$-0.80_{\pm 1.74}$	$-6.89_{\pm 0.86}$	$-5.24_{\pm 2.32}$
	SalLoc-RL/RFT ($\alpha = 30\%$)	$3.80_{\pm 0.39}$	$3.30_{\pm 1.54}$	$4.55_{\pm 0.32}$	$-1.71_{\pm 0.31}$
	DEL ($\alpha = 30\%$)	$-0.97_{\pm 0.40}$	$0.64_{\pm 1.23}$	$-4.26_{\pm 0.32}$	$-0.78_{\pm 0.92}$

A.6 Measuring utility via retain Accuracy

Table 11 presents the retain performance when pairing various localization strategies and unlearning algorithms. The retain performance is measured as the difference between the accuracy of the oracle and unlearned models on the retain set ($\Delta_{\text{retain}} = \text{Oracle}_{\text{retain}} - \text{Unlearn}_{\text{retain.}}$). The experiments are conducted using the ResNet-18 model on the CIFAR-10 dataset with a non-IID forget set consisting of 10% of randomly selected training samples. This table provides an extension of the evaluation metrics shown in Figure 3.

In terms of retain accuracy, the performance of our method is comparable to, or sometimes exceeds, the other methods of comparison. By updating only a small portion of parameters (20% or 30%) as suggested by our localization strategy, unlearning algorithms such as Random Labeling and Reset + Finetuning can achieve the Oracle retain accuracy.

Table 11: **Retain performance (Δ_{retain})** of combining different localization strategies and unlearning algorithms. The retain accuracy values from the unlearned models are provided in $(\cdot)$.

Localization Strategy	**Unlearning Algorithm**		
(α =parameter%)	**Random Label**	**NegGrad+**	**Reset + Finetune**
CritMem($\alpha = 16\%$)	$7.36_{\pm 0.04}$ $(92.63_{\pm 0.09})$	$0.01_{\pm 0.003}$ $(99.98_{\pm 0.005})$	$0.02_{\pm 0.018}$ $(99.97_{\pm 0.04})$
Shallowest($\alpha = 14\%$)	$7.63_{\pm 0.02}$ $(92.36_{\pm 0.04})$	$0.02_{\pm 0.008}$ $(99.98_{\pm 0.02})$	$0.013_{\pm 0.002}$ $(99.98_{\pm 0.004})$
Shallowest($\alpha = 25\%$)	$7.41_{\pm 0.04}$ $(92.58_{\pm 0.09})$	$0.03_{\pm 0.003}$ $(99.96_{\pm 0.04})$	$0.007_{\pm 0.003}$ $(99.99_{\pm 0.006})$
SalUn($\alpha = 20\%$)	$7.76_{\pm 0.04}$ $(92.23_{\pm 0.09})$	$0.05_{\pm 0.005}$ $(99.94_{\pm 0.01})$	$0.001_{\pm 0.001}$ $(99.99_{\pm 0.002})$
SalUn($\alpha = 30\%$)	$6.94_{\pm 0.04}$ $(93.06_{\pm 0.04})$	$0.05_{\pm 0.009}$ $(99.94_{\pm 0.006})$	$0.00_{\pm 0.00}$ $(100.00_{\pm 0.00})$
Ours($\alpha = 20\%$)	$6.97_{\pm 0.02}$ $(93.02_{\pm 0.05})$	$-0.04_{\pm 0.007}$ $(99.95_{\pm 0.01})$	$0.00_{\pm 0.00}$ $(100.00_{\pm 0.00})$
Ours($\alpha = 30\%$)	$6.83_{\pm 0.02}$ $(93.16_{\pm 0.06})$	$-0.04_{\pm 0.008}$ $(99.95_{\pm 0.02})$	$0.00_{\pm 0.00}$ $(100.00_{\pm 0.00})$

A.7 SVHN / ViT and ImageNet-100 / ResNet-50 Experimental Results

The full results corresponding to Figure 4 are detailed in Tables 12 and 13.

Table 12: Comparison to state-of-the-art, including algorithms that update all parameters ("**Full-Parameter**") on **Non-IID** and **IID** forget set when training a **ViT** model on **SVHN** dataset.

		Non-IID Forget Set			**IID Forget Set**		
		Δ_{forget}	Δ_{MIA}	Δ_{test}	Δ_{forget}	Δ_{MIA}	Δ_{test}
Full-parameter Unlearning	Retraining(Oracle)	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
	Fine-tuning	$-11.27_{\pm 0.66}$	$11.22_{\pm 0.65}$	$-0.76_{\pm 0.20}$	$-2.73_{\pm 0.49}$	$-1.05_{\pm 0.42}$	$-0.91_{\pm 0.22}$
	NegGrad+	$-3.45_{\pm 1.65}$	$3.38_{\pm 1.67}$	$2.32_{\pm 0.27}$	$3.22_{\pm 3.58}$	$-6.99_{\pm 3.53}$	$3.75_{\pm 3.53}$
	NegGrad	$-19.75_{\pm 0.74}$	$19.70_{\pm 0.53}$	$0.10_{\pm 0.21}$	$-11.78_{\pm 0.36}$	$8.00_{\pm 0.22}$	$0.58_{\pm 0.21}$
	Random Label	$-5.20_{\pm 1.78}$	$5.13_{\pm 1.81}$	$3.22_{\pm 0.67}$	$-0.96_{\pm 0.54}$	$-2.82_{\pm 0.46}$	$3.69_{\pm 0.47}$
	L1-sparse	$-8.72_{\pm 0.34}$	$8.66_{\pm 0.31}$	$0.24_{\pm 0.19}$	$1.36_{\pm 0.46}$	$-2.43_{\pm 0.39}$	$1.52_{\pm 0.29}$
	IU	$1.57_{\pm 0.28}$	$5.04_{\pm 0.91}$	$3.11_{\pm 0.18}$	$1.45_{\pm 0.36}$	$-5.25_{\pm 0.22}$	$12.41_{\pm 0.21}$
Localized Unlearning	SSD	$2.83_{\pm 1.57}$	$-2.95_{\pm 1.56}$	$3.30_{\pm 0.24}$	$7.26_{\pm 0.88}$	$-11.09_{\pm 0.85}$	$13.26_{\pm 0.74}$
	CritMem-RL ($\alpha = 1\%$)	$-20.77_{\pm 0.27}$	$20.71_{\pm 0.26}$	$-0.45_{\pm 0.18}$	$-11.94_{\pm 0.35}$	$8.16_{\pm 0.22}$	$0.08_{\pm 0.21}$
	Shallowest-RFT ($\alpha = 30\%$)	$5.20_{\pm 2.34}$	$-5.24_{\pm 2.32}$	$3.78_{\pm 1.28}$	$3.09_{\pm 0.90}$	$-6.89_{\pm 0.86}$	$3.56_{\pm 0.59}$
	SalLoc-RFT ($\alpha = 30\%$)	$1.71_{\pm 0.32}$	$-1.71_{\pm 0.31}$	$2.55_{\pm 0.20}$	$0.78_{\pm 0.41}$	$4.55_{\pm 0.32}$	$3.68_{\pm 0.23}$
	DEL ($\alpha = 30\%$)	$0.75_{\pm 0.91}$	$-0.78_{\pm 0.92}$	$0.78_{\pm 0.52}$	$0.46_{\pm 0.043}$	$-4.26_{\pm 0.32}$	$0.89_{\pm 0.29}$

Table 13: Comparison to state-of-the-art, including algorithms that update all parameters ("**Full-Parameter**") and "**Localized**" unlearning on the **IID** forget set when training on **ImageNet-100** with **ResNet-50**. DEL outperforms the state-of-the-art across all unlearning metrics while having the best test accuracy among all localized unlearning methods.

		$\Delta_{\text{forget}} \downarrow$	$\Delta_{\text{MIA}} \downarrow$	$\Delta_{\text{test}} \downarrow$
	Retraining (Oracle)	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
Full-Parameter Unlearning	Fine-tuning	$-6.96_{\downarrow \pm 1.33}$	$6.34_{\downarrow \pm 1.18}$	$0.54_{\downarrow \pm 0.95}$
	NegGrad+	$-3.18_{\pm 1.95}$	$2.54_{\pm 1.52}$	$5.09_{\pm 1.64}$
	NegGrad	$5.13_{\pm 8.05}$	$-5.65_{\pm 8.14}$	$19.68_{\pm 6.60}$
	Random Label	$5.18_{\pm 1.59}$	$-5.49_{\pm 1.03}$	$5.96_{\pm 1.10}$
	L1-sparse	$-5.58_{\pm 1.32}$	$4.76_{\pm 0.94}$	$1.06_{\pm 0.98}$
Localized Unlearning	SSD	$-14.11_{\downarrow \pm 1.96}$	$13.71_{\pm 1.80}$	$5.44_{\pm 1.37}$
	Shallowest-RFT ($\alpha = 30\%$)	$-1.69_{\pm 2.41}$	$2.36_{\pm 2.26}$	$11.72_{\pm 1.50}$
	SalLoc-RFT ($\alpha = 30\%$)	$1.36_{\pm 2.01}$	$-2.19_{\pm 1.73}$	$6.09_{\pm 0.98}$
	DEL ($\alpha = 30\%$)	$\mathbf{0.78_{\pm 1.55}}$	$\mathbf{-1.74_{\pm 1.35}}$	$\mathbf{5.20_{\pm 1.08}}$

A.8 New evaluation metric

We employ a new metric to enable a more fine-grained comparison of unlearning methods. This metric is defined as the proportion of samples that are misclassified by the oracle model but correctly classified by the unlearned model, normalized by the total number of samples. Consider the following example: the ground-truth is [1, 0, 0, 1, 1], an oracle predicts [1, 0, 0, 0, 0] and the unlearned model predicts [0, 0, 1, 1, 1], the value of the new evaluation metric is 2/5=0.4 while the difference between the oracle accuracy and unlearning accuracy is 0. The accuracy metric in this example is misleading because the oracle and unlearning models misclassify totally different subsets of samples, but the new metric does not suffer from this issue.

As shown in the following Table 14 and Table 15, DEL outperforms the baseline methods across different datasets (CIFAR10, SVHN and ImageNet100), models (ResNet18, ResNet50, and ViT), forget set types (IID and Non-IID), and forget set sizes (10%, 30%, and 60% of the training set) according to this metric as well. A lower value for this metric indicates better performance, as an effective unlearning algorithm should forget the correct labels of samples that the oracle model has also forgotten.

Table 14: New evaluation metric on CIFAR10/ResNet18 - IID and Non-IID settings (lower is better)

Method	**IID-10%**	**IID-30%**	**IID-60%**	**Non-IID**
Retraining (Oracle)	00.00	00.00	00.00	00.00
Fine-tuning	6.35	8.29	9.93	13.60
L1-sparse	6.85	8.51	10.90	11.73
Relabel	12.31	15.83	18.39	22.29
IU	11.61	15.35	17.02	19.41
NegGrad+	8.09	7.58	9.97	19.65
Shallowest-RL	12.62	14.66	18.00	23.31
CritMem-RL	12.94	15.82	18.73	22.85
Saloc-RL	13.28	15.65	18.85	21.89
DEL	**5.46**	**6.90**	**8.97**	**10.55**

Table 15: New evaluation metric on SVHN/ViT and ImageNet100/ResNet50 settings (lower is better)

Method	**SVHN / ViT (Non-IID)**	**ImageNet100 / ResNet50 (IID)**
Retraining (Oracle)	00.00	00.00
Fine-tuning	13.17	12.70
L1-sparse	11.81	12.25
Relabel	12.50	9.90
GA	20.69	15.44
NegGrad+	9.18	11.76
SSD	11.82	19.41
Shallowest-RFT/RL	6.15	14.50
Saloc-RL/RFT	7.26	9.88
DEL	**7.08**	**8.56**

Table 16: Comparison of methods on Subclass Unlearning with different forget set sizes (CIFAR-10/ResNet-18)

	Method	**Subclass Unlearning** (forget set: 60%)			**Subclass Unlearning** (forget set: 30%)		
		$\mathbf{\Delta_{forget}} \downarrow$	$\mathbf{\Delta_{test}} \downarrow$	$\mathbf{\Delta_{mia}} \downarrow$	$\mathbf{\Delta_{forget}} \downarrow$	$\mathbf{\Delta_{test}} \downarrow$	$\mathbf{\Delta_{mia}} \downarrow$
Full-parameter Unlearning	Retraining (Oracle)	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
	Fine-tuning	$-3.63_{\pm 1.69}$	$-3.39_{\pm 1.51}$	$3.21_{\pm 2.12}$	$-2.11_{\pm 1.26}$	$-2.22_{\pm 0.84}$	$1.55_{\pm 0.95}$
	L1-sparse	$-2.30_{\pm 1.75}$	$-2.40_{\pm 1.61}$	$1.80_{\pm 1.95}$	$-1.79_{\pm 0.34}$	$-1.50_{\pm 0.30}$	$1.34_{\pm 0.26}$
	Relabel	$-2.76_{\pm 0.59}$	$10.32_{\pm 0.54}$	$2.20_{\pm 0.86}$	$-2.45_{\pm 0.27}$	$7.70_{\pm 0.82}$	$1.63_{\pm 0.20}$
	IU	$-5.47_{\pm 0.45}$	$7.96_{\pm 0.49}$	$3.50_{\pm 0.71}$	$-9.30_{\pm 0.24}$	$7.20_{\pm 0.06}$	$7.45_{\pm 0.18}$
	NegGrad+	$2.19_{\pm 1.29}$	$3.43_{\pm 1.35}$	$-3.03_{\pm 1.55}$	$1.28_{\pm 0.30}$	$2.39_{\pm 0.62}$	$-2.05_{\pm 0.97}$
Localized Unlearning	Shallowest-RL ($\alpha = 25\%$)	$-5.39_{\pm 0.59}$	$8.93_{\pm 0.69}$	$5.17_{\pm 0.99}$	$1.71_{\pm 0.54}$	$10.84_{\pm 0.40}$	$-2.15_{\pm 0.46}$
	CritMem-RL ($\alpha = 23\%, 20\%$)	$-2.87_{\pm 1.12}$	$10.66_{\pm 0.90}$	$2.46_{\pm 1.09}$	$-2.19_{\pm 0.43}$	$7.79_{\pm 0.26}$	$1.31_{\pm 0.45}$
	Saloc-RL ($\alpha = 30\%$)	$-3.10_{\pm 0.72}$	$10.48_{\pm 0.71}$	$2.68_{\pm 0.81}$	$-2.51_{\pm 0.62}$	$7.85_{\pm 0.29}$	$1.33_{\pm 0.54}$
	DEL ($\alpha = 30\%$)	$\mathbf{-0.18_{\pm 0.62}}$	$\mathbf{0.93_{\pm 0.67}}$	$\mathbf{-0.35_{\pm 1.10}}$	$\mathbf{-0.29_{\pm 0.39}}$	$\mathbf{1.50_{\pm 0.12}}$	$\mathbf{-0.29_{\pm 0.30}}$

Table 17: Computational time of localization strategies (CIFAR-10/ResNet-18, 5k–15k–30k)

Strategy / Forget size	**10% (5000 samples)**	**30% (15000 samples)**	**60% (30000 samples)**
CritMem	21.33 hours	62.40 hours	125.33 hours
SalLoc	2.05 seconds	4.82 seconds	9.31 seconds
Shallowest	0.56 milliseconds	0.56 milliseconds	0.56 milliseconds
DEL	2.19 seconds	5.71 seconds	10.09 seconds

Table 18: Continual unlearning on CIFAR-10 / ResNet-18(class $2 \to 5 \to 3 \to 4$)

Method	$\mathbf{\Delta_{forget}} \downarrow$	$\mathbf{\Delta_{test}} \downarrow$	$\mathbf{\Delta_{mia}} \downarrow$
Retraining (Oracle)	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
Fine-tuning	$-10.80_{\pm 0.31}$	$-2.14_{\pm 0.26}$	$6.09_{\pm 0.30}$
L1-sparse	$-2.66_{\pm 0.51}$	$-2.27_{\pm 0.29}$	$-2.64_{\pm 0.28}$
Shallowest-RFT ($\alpha = 25\%$)	$2.40_{\pm 0.77}$	$5.74_{\pm 0.32}$	$-13.44_{\pm 1.48}$
Saloc-RFT ($\alpha = 30\%$)	$5.71_{\pm 0.68}$	$4.92_{\pm 0.50}$	$-4.88_{\pm 0.88}$
DEL($\alpha = 30\%$)	$\mathbf{0.73_{\pm 0.25}}$	$\mathbf{1.33_{\pm 0.28}}$	$\mathbf{0.14_{\pm 0.41}}$