7 Appendix

This Appendix contains the following sections:

- Section 8: Limitations and Future Work
- Section 9: Broader Impact
- Section 10: Experimental Details and Pseudocode
- Section 11: Formal Description of Metrics
- Section 12: Membership Inference Attacks: Description and Additional Findings
- Section 13: Ablations and Sensitivity Analysis
- Section 14: Additional Results for Removing Biases (RB)
- Section 15: Additional Results for Resolving Confusion (RC)
- Section 16: Additional Results for User Privacy (UP)

8 Limitations and Future Work

SCRUB has shown impressive results in terms of being consistently a top-performer in terms of unlearning, with a minimal drop in performance compared to previous works. However, SCRUB has limitations that we hope future work will address.

A significant step for future work is to develop theoretical guarantees for the gains provided by our methods. We opted to focus on an empirical approach for the following reasons: First, while theoretical guarantees abound for linear models, deep networks pose additional significant challenges. Second, methods accompanied with theoretical guarantees suffer from practical limitations with respect to accuracy and/or scalability. For these reasons we opted to approach the problem from a practical standpoint, pushing the envelope by developing unlearning algorithms which are top performers across many important different application scenarios, different evaluation metrics, different architectures and datasets. We look forward to future work that strives to strike a compromise between effective unlearning, good performance, scalability, and theoretical insights.

Another limitation of SCRUB is the difficulty and instability associated with tuning the min-max objective, as shown in the literature e.g. for GANs. For instance, this can lead to oscillating behaviour, as we show in Figure 6. We remedy this to a large extent in practice by providing a practical algorithm that works well, showing consistently improved results over prior work, but there is room for improvement on this front for future work.

SCRUB's rewinding procedure also has limitations. We find in practice throughout all of our experiments that it can help to substantially increase the success of SCRUB's defense on MIA in scenarios where the forget error obtained by SCRUB at the end of unlearning is 'too high'. However, a different failure case which can also appear is that SCRUB's forget error at the end of training is 'too low'. This can happen due to the way in which we tune hyperparameters, which is designed to not harm the retain performance too much, and thus can in some cases lead to 'premature stopping' before the forget error reaches the same level as a reference point for how high it would be if the model had truly never seen those examples. We highlight that addressing all possible issues that can arise in all scenarios and provide an unlearning algorithm that performs strongly across the board is extremely challenging. The fact that we have observed failure cases for each algorithm, be it SCRUB or other baselines, is indicative of the extensiveness of the experimentation we conducted. Our work has made important strides in designing consistently strong-performing unlearning methods and we look forward to future contributions in this direction.

We hope that future work also continues to push the limits of scalability. We believe that our work constitutes an important step in this direction. However, the datasets and models we consider aren't too large, in order to allow comparisons to previous works that would not be feasible to run for larger scale experiments. An interesting topic of future work is investigating the interplay between SCRUB and other scalable algorithms like NegGrad with increasing amounts of scale.

Another really interesting future work direction is to investigate how different unlearning algorithms interact with different architectures, like Transformers, and loss functions, like self-supervised learning.

9 Broader Impact

While recent advances in deep learning represent exciting opportunities for our community, they also come with great responsibility. As researchers, we are responsible for understanding and mitigating the issues associated with the widespread use of deep learning technology. Machine learning models may carry harmful biases, unintended behaviours, or compromise user privacy. Our work is intended to take a step in addressing these issues via a post-processing 'unlearning' phase that makes progress over previous solutions in practice, as we show through an extensive empirical investigation. However, SCRUB does not come with theoretical guarantees: we can not prove that applying SCRUB completely mitigates those issues in all scenarios, so caution must be taken in practice and proper auditing of machine learning models is critical, as advocated by previous works.

10 Experimental Details and Pseudocode

Algorithm 1 SCRUB

Require: Teacher weights w^o
Require: Total max steps MAX-STEPS
Require: Total steps STEPS
Require: Learning rate ϵ
$w^u \leftarrow w^o$
$i \leftarrow 0$
repeat
  if $i <$ MAX-STEPS **then**
    $w^u \leftarrow$ DO-MAX-EPOCH(w^u)
  end if
  $w^u \leftarrow$ DO-MIN-EPOCH(w^u)
until $i <$ STEPS

Datasets. We have used CIFAR-10 and Lacuna-10 datasets for evaluation purposes. CIFAR-10 consists of 10 classes with 60000 color images of size 32 x 32. In our experiments, the train, test, and validation sizes are 40000, 10000, and 10000 respectively. Lacuna-10 is a dataset derived from VGG-Faces [Cao and Yang, 2015]. We have followed the same procedure described in [Golatkar et al., 2020a] to build Lacuna. We randomly select 10 celebrities (classes) with at least 500 samples. We use 100 samples of each class to form the test-set, and the rest make the train-set. All the images are resized to 32 x 32. We also use CIFAR-100 and Lacuna-100 to pretrain the models. Lacuna-100 is built in a similar way as Lacuna-10, and there is no overlap between the two datasets. We have not applied any data augmentation throughout the experiments.

Small-Scale datasets. We followed the same procedure as decribed in [Golatkar et al., 2020b] to create the small versions of CIFAR-10 and Lacuna-10, namely CIFAR-5 and Lacuna-5. To this end, we take the first 5 classes of each dataset and randomly sample 100 images for each class. We make the train and test sets by sampling from the respective train and test sets of CIFAR-10 and Lacuna-10. We also make 25 samples from each class from the train set to create the validation sets.

Algorithm 2 DO-MAX-EPOCH

Require: Student weights w^u
Require: Learning rate ϵ
Require: Batch size B
Require: Forget set D_f
Require: Procedure NEXT-BATCH
$b \leftarrow$ NEXT-BATCH(D_f, B)
repeat

$$w^u \leftarrow w^u + \epsilon \nabla_{w^u} \frac{1}{|b|} \sum_{x_f \in b} d(x_f; w^u)$$

  $b \leftarrow$ NEXT-BATCH(D_f, B)
until b

Models. We use the same models with the same architectural modifications in [Golatkar et al., 2020a,b]. For All-CNN, the number of layers is reduced and batch normalization is added before each non-linearity. For Resnet, ResNet-18 architecture is used. For small scale experiments, the number of filters is reduced by 60% in each block. For the large-scale experiments, the exact architecture has been used.

Pretraining. Following the previous work for consistency, we apply pretraining. Specifically, for CIFAR datasets, we have pretrained the models on CIFAR-100. For Lacuna, we have pretrained the models on Lacuna-100. We pretrain the models for 30 epochs using SGD with a fixed learning rate of 0.1, Cross-Entropy loss function, weight decay 0.0005, momentum 0.9, and batch size 128.

Baselines. 'Original' is the model trained on the entire dataset D. For 'Retrain', we train the same architecture on D_r, with the same hyperparameters used during training of the original model. For 'Finetune', we fine-tune the 'original' model on D_r for 10 epochs, with a fixed learning rate of 0.01 and weight-decay 0.0005. For 'NegGrad', we fine-tune the 'original' model using the following loss:

$$\mathcal{L}(w) = \beta \times \frac{1}{|D_r|} \sum_{i=1}^{|D_r|} l(f(x_i; w), y_i) - (1-\beta) \times \frac{1}{|D_f|} \sum_{j=1}^{|D_f|} l(f(x_j; w), y_j) \tag{4}$$

Where $\beta \in [0, 1]$. We have tuned β to get a high forget-error while not destroying retain-error. For small-scale experiments, $\beta = 0.95$ and we have trained for 10 epochs, with SGD, 0.01 lr and 0.1 weight-decay. For large-scale experiments $\beta = 0.9999$ and we have trained for 5 epochs, with SGD, 0.01 lr, and 0.0005 weight-deay. Please note that small β result in explosion quickly. For 'CF-k', we freeze the first k layers of the network and finetune the rest layers with D_r. We use the same setting as 'Finetune' baseline. For 'EU-k' we freeze the first k layers, and re-initialize the weights of the remaining layers and retrain them with D_r. As all the models are pretrained on larger datasets, for re-initializing we use the weights of the pretrained models. For 'EU-k' we use the same settings as the 'Retrain' baseline. In both 'EU-k' and 'CF-k' baselines, for both ResNet and All-CNN we freeze all the layers except for the last block of the network. For Resnet the last block is block4 and for All-CNN, the last block of layers is the 9th sequential block. For Bad-T, we follow the specifications given in Chundawat et al. [2022] with possible tuning of the parameters in different settings to get the highest forget-error without damaging retain-error. More specifically, for all models we perform one epoch of unlearning using Adam optimizer, and a temperature scalar of 4. Also, we use the whole retain-set compared to 30% reported in their paper as we empirically observed that using only 30% of retain-set for Bad-T yields high test errors.

Algorithm 3 DO-MIN-EPOCH

Require: Student weights w^u
Require: Learning rate ϵ
Require: Batch size B
Require: Retain set D_r
Require: Procedure NEXT-BATCH
  $b \leftarrow$ NEXT-BATCH(D_r, B)
  repeat
    $w^u \leftarrow w^u - \epsilon \nabla_{w^u} \frac{1}{|b|} \sum_{(x_r, y_r) \in b} \alpha d(x_r; w^u) + \gamma l(f(x_r; w^u), y_r)$
    $b \leftarrow$ NEXT-BATCH(D_r, B)
  until b

SCRUB pseudocode and parameters. We train SCRUB using Algorithm 1. Throughout the experiments, we tune the parameters to get a high forget-error while retaining the retain error of the original model. We use the same optimizer for both min and max steps. We observed that for small-scale settings 'Adam' optimizer works better, while in large-scale settings both 'Adam' and 'SGD' could be used. For all experiments, we initialize the learning rate at 0.0005 and decay it by 0.1 after a number of min and max steps. Decaying the learning rate is crucial to control the oscillating behaviour of our min and max optimization. We apply a weight decay of 0.1 for small-scale setting and 0.0005 for large scale experiments, with a momentum of 0.9. Finally, we use different batch sizes for the forget-set and the retain-set to control the number of iteration in each direction, i.e the max and the min respectively. We report these in Table 3.

System specification. For scale-up experiments, the code is executed in Python 3.8, on an Ubuntu 20 machine with 40 CPU cores, a Nvidia GTX 2080 GPU and 256GB memory.

11 Formal Description of Metrics

In this section, we give more details and mathematical definitions of the metrics that we use throughout the paper. We first mathematically define the forget, retain and test errors, and then other application-dependent metrics, for Resolving Confusion (RC) and User Privacy (UP).

Forget, retain and test errors Here, we define the retain error, forget error and test error. Let $\mathcal{D}_r$, $\mathcal{D}_f$ and $\mathcal{D}_t$ denote the retain and forget portions of the training dataset, and a test dataset of heldout examples, respectively. We define error (Err) as follows:

$$Err(\mathcal{D}) = 1 - \frac{1}{|\mathcal{D}|} \sum_{(x_i, y_i) \in \mathcal{D}} \mathbb{1}[\arg\max(f(x_i; w)) == y_i] \tag{5}$$

model	dataset	unlearning-type	forget-set bs	retain-set bs	max steps	min steps
ResNet	CIFAR-10	class	512	128	2	3
	CIFAR-10	selective	16	64	5	5
	Lacuna-10	class	128	128	5	5
	Lacuna-10	selective	32	32	4	4
	CIFAR-5	selective	32	32	10	10
	Lacuna-5	selective	32	32	5	10
All-CNN	CIFAR-10	class	512	256	3	4
	CIFAR-10	selective	16	64	5	5
	Lacuna-10	class	32	32	4	4
	Lacuna-10	selective	8	32	2	4
	CIFAR-5	selective	16	32	5	10
	Lacuna-5	selective	32	32	5	10

Table 3: SCRUB's hyperparameters for each experiment

where f, parameterized by w is the neural network model (comprised of a feature extractor followed by a softmax classifier layer), $\arg\max(f(x_i; w))$ is the label that the model thinks is most likely for example x_i, and $\mathbb{1}[x]$ is the indicator function that returns 1 if x is True and 0 otherwise.

Based on the above, the retain error, forget error and test error are computed as $Err(\mathcal{D}_r)$, $Err(\mathcal{D}_f)$ and $Err(\mathcal{D}_t)$, respectively.

Metrics for Unlearning for Resolving Confusion (RC) We now define the class confusion metrics inspired by Goel et al. [2022]. Specifically, we explore a scenario where the forget set has confused labels (e.g. for two classes A and B, examples of A are labelled as B, and vice versa). The idea here is that, because mislabelled examples are only present in the forget set, successful unlearning (removing the influence of the forget set) would lead to a model that is not at all confused between classes A and B.

In more detail, the setup we follow is: 1) We first mislabel some portion of the training dataset (we mislabelled examples between classes 0 and 1 of each of CIFAR-5 and Lacuna-5 in our experiments), 2) train the 'original model' on the (partly mislabelled) training dataset (it has mislabelled examples for classes 0 and 1 but correct labels for the remaining classes), 3) perform unlearning where the forget set contains all and only the confused examples. Given this, the goal for the unlearning algorithm is to resolve the confusion of the original model.

We consider the following metrics (using terminology consistent with Goel et al. [2022]). They are presented in order of decreasing generality, and increasing focus on measuring degrees of confusion between the two classes considered.

- **Error** (e.g. test error, retain error, forget error). This counts all mistakes, so anytime that an example of some class is predicted to be in any other class, it will be counted. These are the same metrics that we use for the rest of the paper (see Equation 5). For test and retain error, lower is better, whereas for forget error, higher is better.
- **Interclass Confusion IC-ERR** (e.g. IC test error, IC retain error). This counts only mistakes that involve examples from the confused classes A and B. Specifically, it counts instances of any example of class A being predicted to be in *any* other class , and similarly for class B. Compared to Error, this metric is more focused towards understanding the result of the introduced confusion, since it only considers cases that relate to the confused classes. A successful unlearning method would make no such errors, so lower is better for each of IC test error and IC retain error.
- **FGT-ERR** (e.g. Fgt test error, Fgt retain error). This metric counts only misclassification *between the confused classes* A and B. Here, a mistake of an example of class A (or B) being predicted to be in class other than A or B will not be counted. Only mistakes of an example of class A being predicted to be in class B, and vice versa, are counted. **This is the most focused metric that directly measures the amount of remaining confusion between the two classes in question**. A successful unlearning method would make no such errors, so lower is better for each of Fgt test and Fgt retain.

More formally, Error is the same as defined in Equation 5. Let us now mathematically define IC-ERR and FGT-ERR. We denote by $C^{w,\mathcal{D}}$ the confusion matrix for model parameterized by w on the dataset $\mathcal{D}$, and let $\mathcal{D}_A$ denote the part of the dataset $\mathcal{D}$ that belongs to class A. So, for example $\mathcal{D}_{r_A}$ denotes the part of the retain set $\mathcal{D}_r$ that belongs to class A, and the entry $C^{w,\mathcal{D}}_{A,B}$ of the confusion matrix stores the number of times that a sample belonging to class A was (mis)classified as belonging to class B in the dataset $\mathcal{D}$ by the model parameterized by w. Then, we have:

$$\text{IC-ERR}(\mathcal{D}, A, B; w) = \frac{\sum_k C^{w,\mathcal{D}}_{A,k} + \sum_{k'} C^{w,\mathcal{D}}_{B,k'}}{|\mathcal{D}_A| + |\mathcal{D}_B|} \tag{6}$$

where $k \neq A, k' \neq B$.

So, for example, the 'IC test error' column in our tables is computed via IC-ERR$(\mathcal{D}_t, 0, 1; w)$, where $\mathcal{D}_t$ denotes the test set, and 0 and 1 are the two classes confused in our experiments. Analogously, 'IC retain error' is computed as IC-ERR$(\mathcal{D}_r, 0, 1; w)$

Finally:

$$\text{FGT-ERR}(\mathcal{D}, A, B; w) = C^{w,\mathcal{D}}_{A,B} + C^{w,\mathcal{D}}_{B,A} \tag{7}$$

That is, FGT-ERR only measures the misclassification between the two confused classes A and B. So, for example, the 'Fgt test error' in our tables is computed as FGT-ERR$(\mathcal{D}_t, 0, 1; w)$ and analogously 'Fgt retain error' is computed as FGT-ERR$(\mathcal{D}_r, 0, 1; w)$.

User Privacy (UP) Metrics Please see the next section for full details for each of the two Membership Inference Attacks (MIAs) that we use and experimental results.

12 Membership Inference Attacks: Description and Additional Findings

As mentioned in our paper, we utilize two different MIAs: 1) a 'Standard MIA' that is similar to the ones typically used in unlearning papers (but far from the state-of-the-art of MIAs used by privacy and security colleagues), and 2) the first, to our knowledge, adaptation of the state-of-the-art LiRA attack [Carlini et al., 2022] to the framework of unlearning ('LiRA-for-unlearning' MIA).

In this section, we use the term 'target model' to refer to the model that is being attacked and the term 'target example' to refer to an example whose membership status ('in' or 'out') the attacker tries to predict, based on the 'behaviour' (e.g loss value) of that example under the 'target model'. In both attacks that we consider, the target model is the unlearned model, and target examples are either forget set ('in') or test set ('out') examples. The unlearning algorithm successfully defends an MIA if the attacker can't tell apart examples that were unlearned (forget set examples) from examples that were truly never seen.

12.1 Standard MIA

Returning to our previous notation, let $l(f(x; w^u), y)$ denote the cross-entropy loss of the unlearned model (a deep network f with weights w^u) on example x with label y. We abbreviate this as $l(x, y)$ from now on; dropping the dependence on f and w^u.

The attacker is a binary classifier that takes as input loss values, coming from either the forget set $\mathcal{D}_f$ or a held-out test set $\mathcal{D}_t$, and predicts whether the example whose loss value was presented was in the training set of the original model. We train this attacker via supervised learning on a class-balanced labelled training set for this binary problem: $\mathcal{D}^b_{train} = \{(l(x_i, y_i), y^b_i)\}$ where each x_i is an example coming either from D_f or D_t, and its binary label y^b_i is defined as being 0, if $x_i \in \mathcal{D}_t$ and 1 if $x_i \in \mathcal{D}_f$. Once the binary classifier attacker is trained, we use it to make predictions for a held-out evaluation set of the binary problem: $\mathcal{D}^b_{eval} = \{(l(x_i, y_i), y^b_i)\}$ that is also balanced between examples coming from $\mathcal{D}_f$ and $\mathcal{D}_t$, but is disjoint from $\mathcal{D}^b_{train}$.

The attacker succeeds if it achieves high accuracy on $\mathcal{D}^b_{eval}$, meaning that it can tell apart examples that were part of the original training set from those that weren't, which marks a defeat for the unlearning model in terms of this metric, since it has 'left traces behind' (in this case, in terms of loss values) and leaks information about membership in the forget set. We consider that an optimal

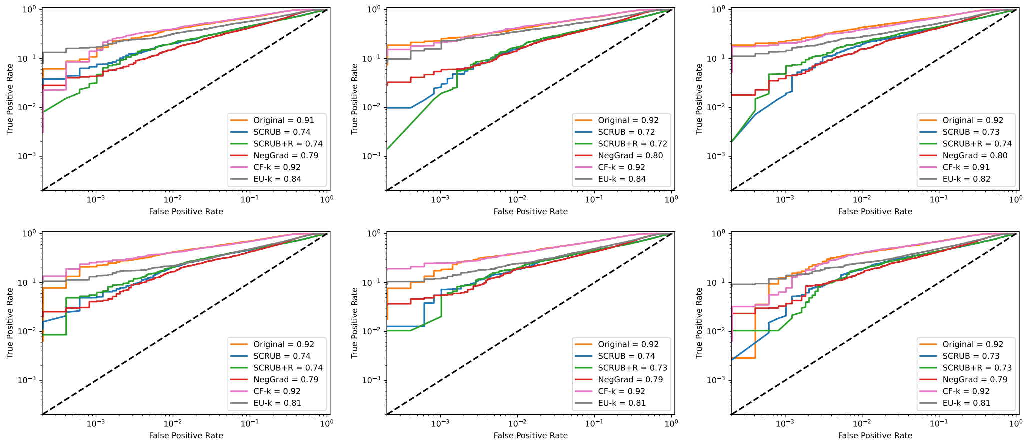


Figure 4: ROC curves for the strong LiRA-for-unlearning attack (Area Under the Curve (AUC) is also reported in the legend, for each unlearning method). Different subplots correspond to different target models (we trained the target unlearned model 6 times for each unlearning method, using different random seeds, and different forget sets). Positives are examples in the forget set, and negatives in the test set. A true positive means that the attacker correctly identified that an example was forgotten, whereas a false positive means that it incorrectly predicted that a test example was forgotten. We are primarily interested in the area of small False Positive Rate [Carlini et al., 2022] and a good unlearning method is associated with a smaller True Positive Rate, i.e. fewer successes for the attacker, especially in the region of interest. **We observe that SCRUB(+R) defends the strong LiRA-for-unlearning attack more successfully than the other baselines.**

defense against this MIA corresponds to a 50% attack accuracy; that is, no better than randomly guessing whether an example had been trained on. In principle, the Retrain oracle should defend optimally: it in fact did not train on the forget set, so the forget and test sets are simply two different held-out sets for this model whose loss values should generally be indistinguishable from each other if these sets are identically-distributed. We find in our experiments, presented in the main paper, that SCRUB+R is able to defend this MIA comparably to Retrain, and outperforms the other baselines in its ability to consistently do so.

Experimental details In practice, if the distribution of the forget set and the test set are very different from each other, their loss values will be very distinguishable. This means that the binary classifier can tell them apart easily, but without having truly learned to infer membership in the training dataset. This makes the attacker's evaluation unreliable. To circumvent this problem, we ought to pick the held-out test set from the same distribution. More specifically, if the forget set is examples from the 'cat' class of CIFAR10 dataset, we use the same class for our held-out test set.

In our experiments, we clip the loss values to a range between [-400, +400] to remove anomalies. Also, we use the default LogisticRegression() classifier of the Python's scikit-learn library as our attack model, and perform a cross-validation with 5 random splits. We report the average accuracy of the evaluation part of each of the 5 folds as the MIA score. Ideally (for a perfect defense), this score is closest to 50%, indicating that the attacker fails to tell apart the forget set from the test set.

We present additional results for the Standard MIA attack in Section 16.

12.2 LiRA-for-unlearning attack

In the standard privacy setting, the LiRA attacker [Carlini et al., 2022] trains a large number of 'shadow models' [Shokri et al., 2017], for which it controls which examples are in the training set each time (by construction). To then predict the membership status of a 'target example', it estimates two Gaussian distributions, using the shadow models: the distribution of confidences of that example under shadow models that trained on it, and the distribution of its confidences under shadow models that didn't. Then, it computes the confidence of the target example under the target model and it predicts that the target example was ‘in’ if the likelihood of the target confidence under the former Gaussian is larger than that under the latter Gaussian.

Adapting LiRA to the framework of unlearning is not trivial, and we are not aware of this done in any previous work. **We propose the first, to our knowledge, adaptation of LiRA for unlearning**. This is a strong attack where we allow the attacker knowledge of the unlearning algorithm. Concretely, for each shadow model, the attacker also produces a ‘shadow unlearned’ model by applying the given unlearning algorithm several times, using a large number of forget sets (similar to Chen et al. [2021]). Now, for each ‘target example’, this setup allows the attacker to estimate a different pair of Gaussians: the distribution of (confidences of) that target example under models where it was *forgotten*, and as before, under models where it was not seen. The attacker then computes the confidence of the target example under the target model, and predicts the example was forgotten if its likelihood under the former is larger than under the latter.

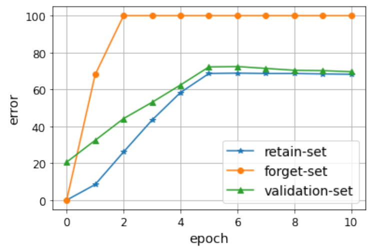
Experimental setup: overview We run our LiRA-for-unlearning attack on selective unlearning on CIFAR-10, for the scenario where the forget set has size 200 and comes from class 5.

Attacker: For the attacker, we first train 256 ‘shadow original’ models on random splits of half of the CIFAR-10 training set. Let D denote the original dataset. To train each shadow model, we split D in half, and use one half to as the ‘training set’ and the other half as the ‘test set’ of that particular shadow model. Then, for each of these ‘shadow original’ models, we run unlearning on 10K different forget sets (for each unlearning method). Specifically, the forget set is a random subset of 200 examples of class 5, sampled from the training set of the corresponding ‘shadow original’ model. After this procedure, for every example in class 5, we select 256 associated shadow models when it was not included in training, and 256 shadow models when it was unlearned (i.e. it was in the forget set). After this, the LiRA attack proceeds as normal, where we take each in / out distribution and apply a likelihood ratio test on an unknown example to infer membership.

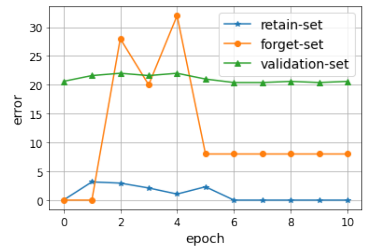
Figure 5: Sensitivity of SCRUB to γ and α. To create this plot, we ran SCRUB many times for different values of γ ([0.1, 0.5, 1, 2, 3]) and α ([0.1, 0.5, 1, 2, 3]). The x-axis is represents combinations of these values. t-error, f-error and r-error refer to test, forget and retain error, respectively. We find that SCRUB is not very sensitive to these hyperparameters: the retain error remains low across values, and there are several different settings to these hyperparameters for which we can obtain the desired results for test and forget errors too.

Defender: We next train the target model that LiRA-for-unlearning will attack. For this, we begin by training the ‘original’ model, on (a random split of) half of D. Then, we apply the given unlearning algorithm on a randomly-sampled forget set, which is a subset of the original model’s training set of size 200, coming from class 5, in the same way as was done by the shadow models.

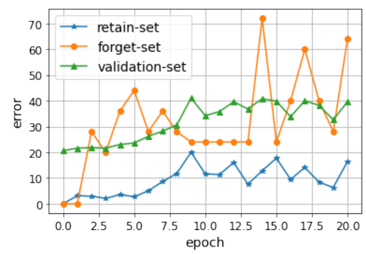
Implementation note We run this attack at a much larger scale than the remaining experiments of the paper (we run 10K unlearning runs, on different forget sets, for *each* of the 256 ‘shadow original’ models). We do this because the strength of the attacker is heavily dependent on the number of shadow original/unlearned models, and we wanted to benchmark our baselines against a very strong attacker. Therefore, to allow better scaling, instead of implementing SCRUB+R as rewinding, we implement it through filtering runs of SCRUB that don’t satisfy the condition of SCRUB+R that the forget set error should be close to the validation error (where, as explained in the main paper, this refers to the validation set that is constructed to have the same distribution as the forget set; containing examples of only the same class as the one in the forget set). We used 0.1 as our threshold.



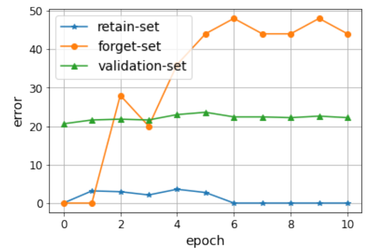
(a) Only max steps (Equation 1).



(b) SCRUB with insufficient max steps.



(c) SCRUB with too many max steps.



(d) SCRUB.

Figure 6: Illustration of training dynamics of SCRUB variants, on CIFAR-5 with a ResNet model. Performing the right interleaving of *min-steps* and *max-steps* is important for achieving a good balance between high forget error and low retain and validation errors.

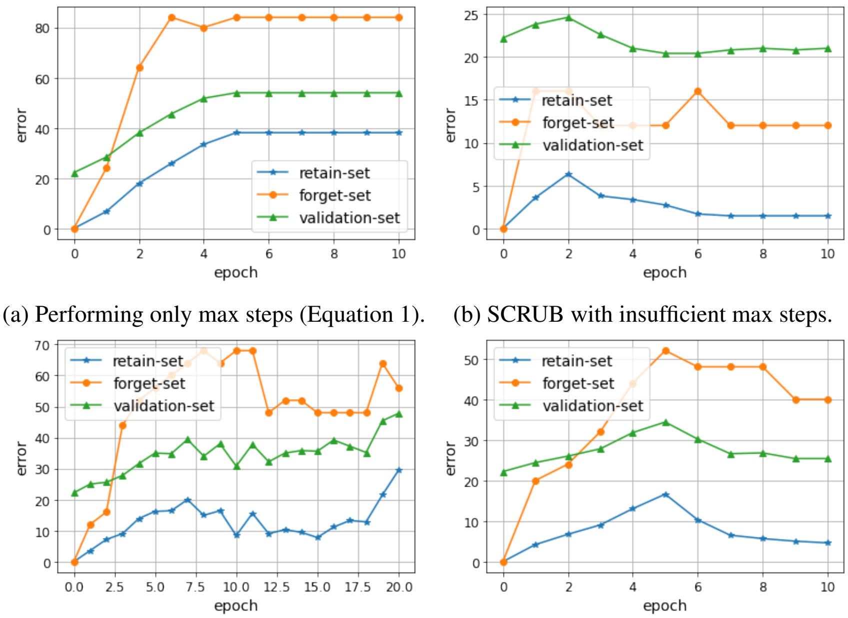
Computing the 'confidence' Consistent with [Carlini et al., 2022], the confidence of an example (x, y) (where y denotes the ground-truth class label) is defined as $\mathrm{softmax}(f(x))[y]$. In words, the confidence is the softmax probability of the *correct class*. Following Carlini et al. [2022], we apply logit-scaling to each confidence, to make their distributions Gaussian.

Conclusions and findings Figure 4 plots the ROC curve, showing the False Positive Rate and True Positive Rate of the attacker, in log-log scale. Different subplots correspond to different target unlearned models, each of which was trained with a different random seed, and different retain/forget set split. A successful defense is associated with a smaller Area Under the Curve (AUC); meaning fewer True Positives for the attacker. Carlini et al. [2022] however advocate that the AUC is not a good indicator of the attacker's strength and, instead, they argue that we should primarily consider the region of the ROC curve associated with very small False Positive Rates. We observe that, especially in that region, SCRUB(+R) is the strongest method in terms of defending our LiRA-for-unlearning attack (and also we observe that SCRUB(+R) has the best AUC too). The improved NegGrad baseline that we also proposed in this paper is also a strong model in terms of defending this attack. We found that CF-k is not able to improve the privacy of the original model in most cases, while EU-k can sometimes improve but only slightly, and not reliably.

Limitations To stay consistent with previous work on unlearning, as mentioned previously, we turn off data augmentations. Consequently, the 'original model' (before unlearning is applied) has overfitted more than a state-of-the-art CIFAR model would. Indeed, as can be seen from Figure 4, the 'Original' model has poor privacy (the attacker has a high True Positive Rate). We note that this is the first, to our knowledge, investigation of privacy of unlearning algorithms using strong MIAs, and we hope that future work continues to investigate increasingly more realistic scenarios with models closer to the state-of-the-art, and considers unlearning on original models of varying degrees of privacy and generalization ability.

13 Ablations and Sensitivity Analysis

In this section, we illustrate the training dynamics of SCRUB and the importance of different design choices. As a reminder, the student is initialized from the teacher and subsequently undergoes an alternating sequence of *max-steps* and *min-steps*; the former encouraging the student to move far



(a) Performing only max steps (Equation 1). (b) SCRUB with insufficient max steps.

(c) SCRUB with too many max steps. (d) SCRUB.

Figure 7: Illustration of training dynamics of SCRUB variants, on CIFAR-5 with All-CNN. Performing the right interleaving of *min-steps* and *max-steps* is important for achieving a good balance between high forget error and low retain and validation errors.

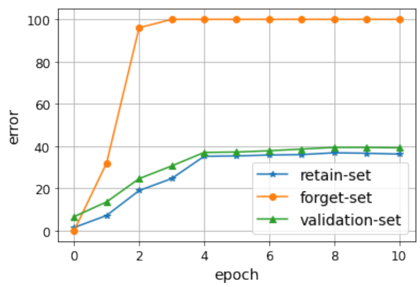
from the teacher on the forget set, and the latter encouraging it to stay close to the teacher on the retain set. We also found it useful to perform a sequence of additional *min-steps* after the alternating sequence. We now explore the effect of these decisions.

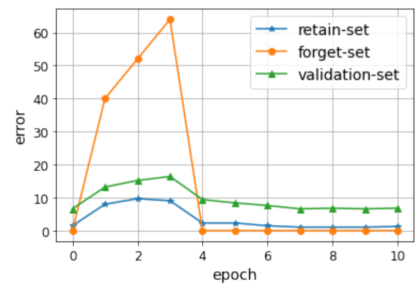
First, we show that performing only *max-steps*, by optimizing Equation 1, is not a good solution. Simply pushing the student away from the teacher on the forget set achieves forgetting but unfortunately also hurts the retain and validation set performance (Figure 6a). Therefore, alternating between *max-steps* and *min-steps* is necessary. However, it is important to find the right balance. For instance, as seen in Figure 6b, performing too few *max-steps* leads to the unwanted consequence of the forget error dropping. On the other hand, removing the final sequence of only *min-steps* is also harmful, as shown in Figure 6c that trains for a larger number of epochs of an equal number of (alternating) *max-steps* and *min-steps* without achieving a good balance at any point throughout the trajectory. On the other hand, SCRUB (Figure 6d) achieves a good balance of high forget error and low retain and validation error simultaneously. We also ablate the cross-entropy term in Equation 3, which provides a small but consistent added protection against degrading performance in Figure 10. We show additional examples of training dynamics (Figures 7, 8, 9).

Finally, we also investigate the sensitivity of SCRUB's results on the γ and α hyperparameters, in Figure 5. We find that SCRUB is not very sensitive to these hyperparameters: the retain error remains low across values, and there are several different settings to these hyperparameters for which we can obtain the desired results for test and forget errors too.

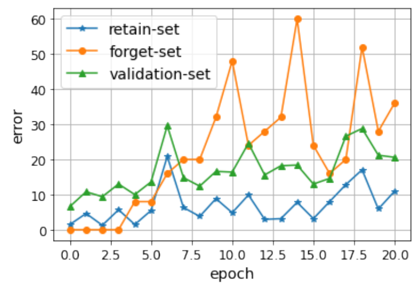
14 Additional Results for Removing Biases (RB)

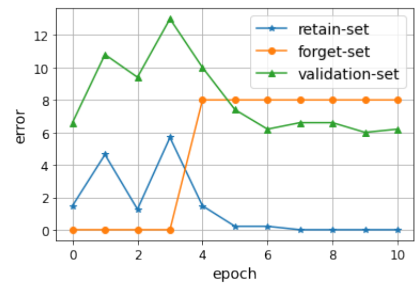
In this section, we provide the results for all scenarios we studied for the **Removing Biases (RB)** application for ResNet and All-CNN, on both CIFAR and Lacuna, for both small-scale and large-scale, for completeness, in Tables 5, 6, 7, 8, 9, 10.





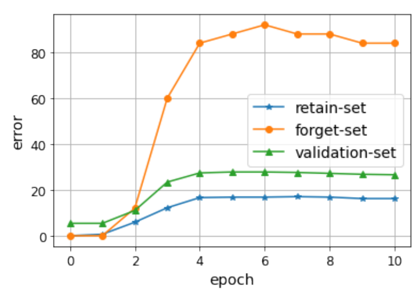
(a) Performing only max steps (Equation 1). (b) SCRUB with insufficient max steps.

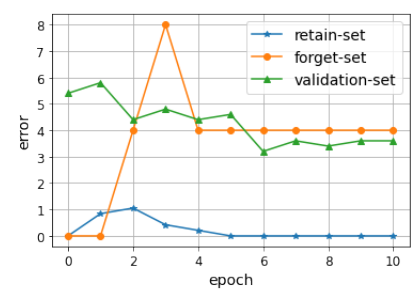




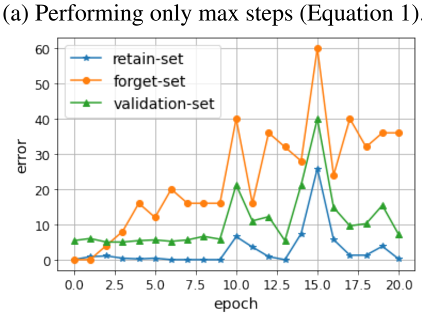
(c) SCRUB with too many max steps. (d) SCRUB.

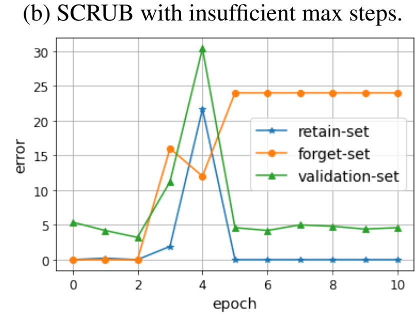
Figure 8: Illustration of training dynamics of SCRUB variants, on Lacuna-5 with ResNet. Performing the right interleaving of *min-steps* and *max-steps* is important for achieving a good balance between high forget error and low retain and validation errors.





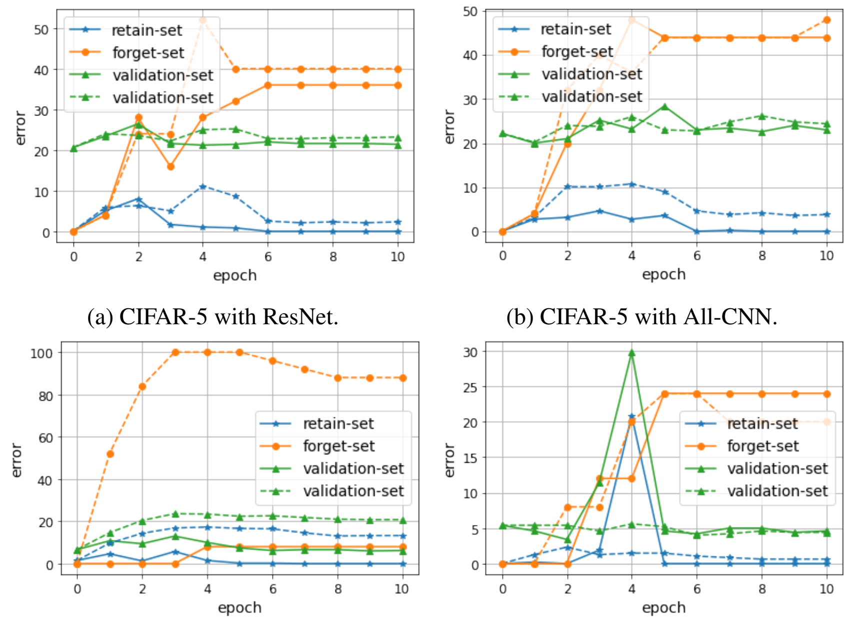
(a) Performing only max steps (Equation 1). (b) SCRUB with insufficient max steps.





(c) SCRUB with too many max steps. (d) SCRUB.

Figure 9: Illustration of training dynamics of SCRUB variants, on Lacuna-5 with All-CNN. Performing the right interleaving of *min-steps* and *max-steps* is important for achieving a good balance between high forget error and low retain and validation errors.



(a) CIFAR-5 with ResNet.

(b) CIFAR-5 with All-CNN.

(c) Lacuna-5 with ResNet.

(d) Lacuna-5 with All-CNN.

Figure 10: Effect of adding the cross-entropy loss in Equation 3. Dashed lines omit cross-entropy while solid lines use it. We find that the addition of cross-entropy offers an additional protection to maintaining the model's performance during the unlearning procedure. This sometimes comes at the cost of smaller forget set error, compared to the forget set error that would have been achieved if cross-entropy was omitted from the loss.

Model	CIFAR-10				Lacuna-10			
	ResNet		All-CNN		ResNet		All-CNN	
	class	selective	class	selective	class	selective	class	selective
Finetune	3.8	3.09	3.33	3.03	1.7	2.03	2.16	2.00
Fisher	0.08	0.07	0.16	0.14	0.08	0.07	0.16	0.15
NegGrad	3.4	2.96	2.30	2.97	1.66	1.5	2.41	2.27
CF-k	3.55	3.17	3.37	2.91	3.42	3.20	3.27	3.11
EU-k	1.41	1.26	1.34	1.20	1.39	1.28	1.32	1.26
Bad-T	19.07	20.44	17.91	17.03	20.05	20.27.	16.32	16.02
SCRUB	7.84	7.41	6.36	5.33	2.17	1.95	2.81	2.48

Table 4: **Scale-up factor**: the fraction of the runtime of retrain from scratch over the runtime of each given unlearning algorithm. That is, a scale-up value of X for an unlearning algorithm means that that algorithm runs X times faster than retrain from scratch.

Model	CIFAR-5						Lacuna-5					
	Test error (↓)		Retain error (↓)		Forget error (↑)		Test error (↓)		Retain error (↓)		Forget error (↑)	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
Retrain	24.9	2.5	0.0	0.0	28.8	5.9	5.8	0.4	0.0	0.0	4.8	3.4
Original	24.2	2.6	0.0	0.0	0.0	0.0	5.7	0.4	0.0	0.0	0.0	0.0
Finetune	24.3	2.4	0.0	0.0	0.0	0.0	5.6	0.3	0.0	0.0	0.0	0.0
Fisher	31.6	3.4	14.0	6.0	4.8	5.2	14.0	3.6	6.7	3.3	6.4	8.3
NTK	24.4	2.6	0.0	0.0	22.4	9.2	5.6	0.5	0.0	0.0	0.0	0.0
NegGrad	25.5	1.1	0.0	0.0	41.3	6.1	6.1	0.7	0.0	0.0	1.3	2.3
CF-k	22.6	1.9	0.0	0.0	0.0	0.0	5.8	0.4	0.0	0.0	0.0	0.0
EU-k	23.5	1.1	0.0	0.0	10.7	2.3	5.9	0.6	0.0	0.0	0.0	0.0
Bad-T	27.73	1.89	5.12	1.56	8.00	8.64	5.00	0.33	0.14	0.10	0.00	0.00
SCRUB	24.2	1.6	0.0	0.0	40.8	1.8	6.2	0.73	0.0	0.0	24.8	5.2

Table 5: **Small-scale** results with ResNet for the **Removing Biases (RB)** application. SCRUB is the top-performer in terms of forgetting with minimal performance degradation.

Model	CIFAR-5						Lacuna-5					
	Test error (↓)		Retain error (↓)		Forget error (↑)		Test error (↓)		Retain error (↓)		Forget error (↑)	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
Retrain	24.36	1.61	0.13	0.28	28.8	9.12	4.6	0.38	0.0	0.0	4.67	6.41
Original	24.08	1.86	0.17	0.38	0.0	0.0	4.53	0.47	0.0	0.0	0.0	0.0
Finetune	23.48	1.91	0.04	0.09	0.0	0.0	9.77	10.76	6.63	13.22	19.33	40.03
Fisher	42.64	6.56	31.83	10.47	15.2	16.83	52.53	13.87	51.09	14.54	39.33	40.43
NTK	24.16	1.77	0.17	0.38	13.6	8.29	4.47	0.47	0.0	0.0	3.33	4.68
NegGrad	26.07	1.21	0.56	0.49	36.00	10.58	5.27	0.76	0.14	0.12	12.00	13.86
CF-k	22.67	1.55	0.00	0.00	0.00	0.00	4.67	0.70	0.00	0.00	0.00	0.00
EU-k	25.87	0.64	3.23	1.69	8.00	6.93	5.20	0.20	0.00	0.00	0.00	0.00
Bad-T	25.87	1.80	9.68	0.45	10.67	4.99	8.87	0.66	2.32	0.79	0.00	0.00
SCRUB	23.88	1.78	0.08	0.12	40.8	8.2	3.87	0.72	0.0	0.0	25.33	4.13

Table 6: **Small-scale** results with All-CNN for the **Removing Biases (RB)** application. SCRUB is the top-performer in terms of forgetting with minimal performance degradation.

Model	CIFAR-10						Lacuna-10					
	Test error (↓)		Retain error (↓)		Forget error (↑)		Test error (↓)		Retain error (↓)		Forget error (↑)	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
Retrain	14.72	0.16	0.0	0.0	100.0	0.0	2.87	0.34	0.0	0.0	99.75	0.56
Original	16.56	0.1	0.0	0.0	0.0	0.0	3.07	0.26	0.0	0.0	0.0	0.0
Finetune	16.41	0.09	0.0	0.0	0.0	0.0	3.02	0.37	0.0	0.0	0.0	0.0
Fisher	26.42	1.41	2.45	0.84	100.0	0.0	3.33	0.54	0.0	0.0	100.0	0.0
NegGrad	17.84	1.46	1.74	2.55	91.26	7.73	3.41	0.17	0.00	0.00	14.90	1.78
CF-k	15.31	0.12	0.00	0.00	0.03	0.01	2.89	0.22	0.00	0.00	0.00	0.00
EU-k	18.73	0.42	0.00	0.00	98.79	0.18	3.19	0.17	0.01	0.02	4.06	0.83
Bad-T	19.56	1.44	11.34	1.82	94.67	6.12	3.37	0.50	1.06	0.47	67.60	24.26
SCRUB	15.73	0.17	0.51	0.02	100.0	0.0	3.69	0.36	0.28	0.23	100.0	0.0

Table 7: **Large-scale, class unlearning** results with ResNet for the **Removing Biases (RB)** application. SCRUB and EU-k are the top-performers in this setting in terms of forgetting with minimal performance degradation. Note, however, that EU-k doesn't perform strongly across the board and in particular performs very poorly in selective unlearning (notice the contrast between EU-k's forget error between Figures 1a and 1b Fisher is also a top-performer in terms of forget error in this setting too, but on CIFAR causes a large degradation in test error, as is often observed for this method.

Model	CIFAR-10						Lacuna-10					
	Test error (↓)		Retain error (↓)		Forget error (↑)		Test error (↓)		Retain error (↓)		Forget error (↑)	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
Retrain	13.97	0.19	0.0	0.0	100.0	0.0	1.59	0.36	0.0	0.0	100.0	0.0
Original	15.56	0.25	0.0	0.0	0.0	0.0	1.56	0.33	0.0	0.0	0.0	0.0
Finetune	15.39	0.22	0.0	0.0	0.0	0.0	1.67	0.44	0.0	0.0	0.0	0.0
Fisher	27.4	2.28	3.66	1.03	99.0	0.0	1.78	0.29	0.0	0.0	89.0	0.0
NegGrad	17.87	0.31	0.58	0.13	87.22	1.67	1.63	0.17	0.00	0.00	6.56	1.13
CF-k	14.99	0.23	0.00	0.00	0.00	0.00	1.48	0.36	0.00	0.00	0.00	0.00
EU-k	15.30	0.69	0.13	0.14	100.00	0.00	1.74	0.45	0.00	0.00	77.19	39.51
Bad-T	16.98	0.40	5.84	0.43	81.93	3.50	2.56	0.09	0.37	0.18	38.65	36.80
SCRUB	15.06	0.14	0.12	0.03	100.0	0.0	2.0	0.4	0.0	0.0	100.0	0.0

Table 8: **Large-scale, class unlearning** results with All-CNN for the **Removing Biases (RB)** application. SCRUB is the top-performer in terms of forgetting with minimal performance degradation.

Model	CIFAR-10						Lacuna-10					
	Test error (↓)		Retain error (↓)		Forget error (↑)		Test error (↓)		Retain error (↓)		Forget error (↑)	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
Retrain	17.4	0.14	0.0	0.0	29.67	3.21	2.7	0.2	0.0	0.0	1.0	1.0
Original	17.36	0.14	0.0	0.0	0.0	0.0	2.73	0.15	0.0	0.0	0.0	0.0
Finetune	17.37	0.11	0.0	0.0	0.0	0.0	2.63	0.12	0.0	0.0	0.0	0.0
Fisher	21.23	0.27	2.88	0.54	3.0	2.65	3.1	0.35	0.0	0.0	0.0	0.0
NegGrad	22.7	0.6	4.1	0.5	53.7	6.8	4.7	0.2	0.9	0.1	13.0	1.0
CF-k	17.4	0.1	0.0	0.0	0.0	0.0	2.7	0.2	0.0	0.0	0.0	0.0
EU-k	21.8	0.2	0.4	0.6	23.7	3.5	2.9	0.1	0.0	0.0	0.0	0.0
Bad-T	23.47	1.57	14.53	1.65	34.67	1.70	7.30	2.20	3.26	1.83	0.33	0.47
SCRUB	18.04	0.2	0.0	0.0	70.33	4.16	3.0	0.0	0.0	0.0	4.67	3.06

Table 9: **Large-scale, selective unlearning** results with ResNet for the **Removing Biases (RB)** application. SCRUB and NegGrad are the top-performers in terms of forgetting, though NegGrad has worse test performance than SCRUB in both cases. Note also that NegGrad isn't as consistent at forgetting across settings as SCRUB.

Model	CIFAR-10						Lacuna-10					
	Test error (↓)		Retain error (↓)		Forget error (↑)		Test error (↓)		Retain error (↓)		Forget error (↑)	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
Retrain	16.47	0.21	0.0	0.0	25.67	2.31	1.6	0.44	0.0	0.0	0.67	0.58
Original	16.43	0.08	0.0	0.0	0.0	0.0	1.53	0.31	0.0	0.0	0.0	0.0
Finetune	16.5	0.18	0.0	0.0	0.0	0.0	1.43	0.21	0.0	0.0	0.0	0.0
Fisher	21.39	1.22	4.0	1.44	13.0	11.27	1.87	0.21	0.01	0.02	0.0	0.0
NegGrad	21.36	0.34	3.23	0.37	45.33	2.89	2.77	0.25	0.40	0.07	8.67	0.58
CF-k	16.29	0.07	0.00	0.00	0.00	0.00	1.53	0.31	0.00	0.00	0.00	0.00
EU-k	17.62	0.61	0.11	0.11	0.33	0.58	1.83	0.47	0.00	0.00	0.00	0.00
Bad-T	22.43	0.37	10.13	0.15	1.67	1.25	4.90	2.10	1.34	1.20	0.67	0.94
SCRUB	16.55	0.11	0.0	0.0	29.33	3.21	2.07	0.31	0.0	0.0	1.67	0.58

Table 10: **Large-scale, selective unlearning** results with All-CNN for the **Removing Biases (RB)** application. SCRUB and NegGrad are the top-performers in terms of forgetting, though NegGrad has worse test performance than SCRUB in both cases. Note also that NegGrad isn't as consistent at forgetting across settings as SCRUB, as can be seen in Figure 2

model	Test error (↓)		Retain error (↓)		Forget error (↑)		IC test error (↓)		IC retain error (↓)		Fgt test error (↓)		Fgt retain error (↓)	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
Retrain	26.67	2.87	0.0	0.0	90.33	1.53	24.0	1.8	0.0	0.0	18.33	4.16	0.0	0.0
Original	41.0	2.09	0.0	0.0	0.0	0.0	56.0	3.04	0.0	0.0	92.0	7.94	0.0	0.0
Finetune	38.13	1.42	0.0	0.0	0.0	0.0	52.0	3.12	0.0	0.0	79.33	10.07	0.0	0.0
NegGrad	36.27	0.42	0.0	0.0	12.67	21.94	47.5	5.27	0.0	0.0	69.0	13.53	0.0	0.0
CF-k	39.6	1.64	0.0	0.0	0.0	0.0	54.83	2.02	0.0	0.0	85.33	7.02	0.0	0.0
EU-k	37.47	1.62	7.33	1.26	43.67	2.08	47.0	4.77	8.33	4.73	63.33	9.71	3.67	2.52
Fisher	44.8	2.36	21.33	3.45	32.0	11.53	51.5	7.47	26.33	9.5	79.0	3.61	20.0	7.94
NTK	32.6	2.51	0.0	0.0	60.33	0.58	37.5	4.0	0.0	0.0	52.0	10.58	0.0	0.0
SCRUB	25.93	3.13	1.08	0.52	96.0	1.73	19.0	3.91	0.0	0.0	19.67	7.51	0.0	0.0

Table 11: Results on CIFAR-5 with ResNet for the **Resolving Confusion (RC)** application. (Confused class 0,1; 50-50 samples). SCRUB is the best-performer by far in terms of eliminating the confusion via unlearning (see the IC error and Fgt error columns), while not hurting performance for other classes (see e.g. the usual Error metrics in the first 3 groups of columns).

model	Test error (↓)		Retain error (↓)		Forget error (↑)		IC test error (↓)		IC retain error (↓)		Fgt test error (↓)		Fgt retain error (↓)	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
Retrain	24.4	2.75	0.0	0.0	90.67	4.04	19.0	1.32	0.0	0.0	11.33	4.62	0.0	0.0
Original	37.07	4.67	1.5	2.6	5.67	9.81	49.0	4.77	6.0	10.39	80.67	12.58	6.0	10.39
Finetune	34.33	3.35	0.0	0.0	3.0	5.2	43.67	7.29	0.0	0.0	67.33	16.04	0.0	0.0
NegGrad	33.53	4.47	0.0	0.0	13.33	21.36	42.33	11.34	0.0	0.0	62.0	22.65	0.0	0.0
CF-k	36.13	4.21	0.0	0.0	0.33	0.58	47.83	5.8	0.0	0.0	76.33	14.43	0.0	0.0
EU-k	51.6	1.0	27.67	3.5	52.67	6.03	59.5	5.22	38.33	6.66	68.67	15.57	19.67	10.41
Fisher	51.93	2.95	35.17	3.92	31.0	11.53	56.83	8.69	31.67	14.01	78.33	15.53	17.67	11.5
NTK	32.2	2.84	0.75	1.3	43.33	14.15	36.67	4.07	3.0	5.2	54.33	9.02	3.0	5.2
SCRUB	25.0	3.14	0.0	0.0	93.33	2.52	26.0	4.44	0.0	0.0	18.0	11.14	0.0	0.0

Table 12: Results on CIFAR-5 with All-CNN for the **Resolving Confusion (RC)** application. (Confused class 0,1; 50-50 samples). SCRUB is the best-performer by far in terms of eliminating the confusion via unlearning (see the IC error and Fgt error columns), while not hurting performance for other classes (see e.g. the usual Error metrics in the first 3 groups of columns).

15 Additional Results for Resolving Confusion (RC)

We show the full results are in Tables 11, 12, 13, 14, 15, 16, 17, 18 for all settings. We observe that across the board, SCRUB is a top-performer on this metric too (see the captions of the individual tables for more details about performance profile).

16 Additional results for User Privacy (UP)

We present Standard MIA results for all settings in Tables 19, 20, 21, 22, 23, 24, 25, 26. We find that SCRUB, especially equipped with its rewinding procedure, is able to consistently have a strong defense against MIAs.

model	Test error (↓)		Retain error (↓)		Forget error (↑)		IC test error (↓)		IC retain error (↓)		Fgt test error (↓)		Fgt retain error (↓)	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
Retrain	6.0	0.2	0.0	0.0	99.67	0.58	7.17	2.57	0.0	0.0	0.0	0.0	0.0	0.0
Original	27.07	3.33	1.67	0.88	4.33	1.53	57.5	6.26	6.67	3.51	108.0	14.18	6.67	3.51
Finetune	18.8	4.26	0.0	0.0	14.67	6.03	37.67	11.15	0.0	0.0	63.67	22.01	0.0	0.0
NegGrad	17.8	2.95	1.67	0.72	55.33	2.08	33.17	5.25	5.33	1.53	56.67	12.9	4.33	1.53
CF-k	22.27	4.31	0.08	0.14	10.67	5.03	46.33	10.97	0.33	0.58	81.67	23.01	0.33	0.58
EU-k	15.27	3.19	0.83	0.38	62.0	12.49	29.33	9.0	2.33	1.53	43.67	16.29	0.33	0.58
Fisher	35.87	3.33	17.75	3.78	27.33	3.79	60.0	5.27	31.0	7.94	109.0	14.53	30.0	7.0
NTK	14.53	5.22	0.0	0.0	51.67	23.18	27.17	11.3	0.0	0.0	43.33	25.32	0.0	0.0
SCRUB	8.47	1.17	0.33	0.14	96.0	1.0	11.33	3.82	1.33	0.58	9.33	1.53	1.33	0.58

Table 13: Results on Lacuna-5 with ResNet for the **Resolving Confusion (RC)** application. (Confused class 0,1; 50-50 samples). SCRUB is the best-performer by far in terms of eliminating the confusion via unlearning (see the IC error and Fgt error columns), while not hurting performance for other classes (see e.g. the usual Error metrics in the first 3 groups of columns). NTK in some cases is able to resolve confusion, but not consistently, and it also suffers from higher Test Error.

model	Test error (↓)		Retain error (↓)		Forget error (↑)		IC test error (↓)		IC retain error (↓)		Fgt test error (↓)		Fgt retain error (↓)	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
Retrain	4.2	0.87	0.0	0.0	100.0	0.0	5.33	2.25	0.0	0.0	0.0	0.0	0.0	0.0
Original	25.47	2.32	5.75	5.63	20.33	25.74	56.17	4.93	23.0	22.54	105.67	8.08	23.0	22.54
Finetune	12.8	2.8	0.0	0.0	23.0	7.94	25.83	7.75	0.0	0.0	39.67	12.74	0.0	0.0
NegGrad	12.8	9.06	2.5	3.12	90.0	6.56	20.33	17.04	5.0	6.24	12.67	11.68	2.67	3.79
CF-k	21.27	1.63	0.58	0.8	9.33	0.58	47.0	4.58	2.33	3.21	82.67	10.12	2.33	3.21
EU-k	17.0	8.91	3.92	3.99	92.33	4.93	35.0	18.26	13.0	11.36	3.67	4.73	0.0	0.0
Fisher	49.6	4.73	39.25	7.45	40.0	9.54	57.67	10.79	42.33	11.59	88.67	11.68	29.67	16.86
NTK	12.87	6.63	2.83	4.91	72.33	12.06	25.5	17.88	11.33	19.63	35.67	24.03	10.0	17.32
SCRUB	3.87	0.7	0.0	0.0	100.0	0.0	4.33	1.26	0.0	0.0	0.0	0.0	0.0	0.0

Table 14: Results on Lacuna-5 with All-CNN for the **Resolving Confusion (RC)** application. (Confused class 0,1; 50-50 samples). SCRUB is the best-performer by far in terms of eliminating the confusion via unlearning (see the IC error and Fgt error columns), while not hurting performance for other classes (see e.g. the usual Error metrics in the first 3 groups of columns).

model	Test error (↓)		Retain error (↓)		Forget error (↑)		IC test error (↓)		IC retain error (↓)		Fgt test error (↓)		Fgt retain error (↓)	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
retrain	18.7	0.07	0.0	0.0	98.57	0.28	14.78	0.18	0.0	0.0	31.33	2.08	0.0	0.0
original	21.86	0.37	0.0	0.0	0.0	0.0	31.23	0.45	0.0	0.0	356.0	11.53	0.0	0.0
finetune	20.85	0.37	0.0	0.0	0.0	0.0	26.75	0.48	0.0	0.0	255.0	10.58	0.0	0.0
NegGrad	23.41	0.32	3.87	0.31	80.07	6.77	41.08	0.6	20.29	1.52	46.0	8.72	0.67	1.15
CF-k	20.93	0.38	0.0	0.0	0.0	0.0	27.27	0.76	0.0	0.0	267.33	16.17	0.0	0.0
EU-k	20.03	0.19	0.25	0.08	95.55	0.54	17.85	0.67	0.18	0.03	53.0	7.94	3.33	2.31
SCRUB	18.01	0.18	0.02	0.01	95.45	0.26	15.07	0.99	0.04	0.03	30.33	3.79	0.33	0.58

Table 15: Results on CIFAR-10 with ResNet for the **Resolving Confusion (RC)** application. (Confused class 0,1; 2000-2000 samples).

model	Test error (↓)		Retain error (↓)		Forget error (↑)		IC test error (↓)		IC retain error (↓)		Fgt test error (↓)		Fgt retain error (↓)	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
retrain	16.43	0.03	0.0	0.0	98.42	0.15	14.37	0.24	0.0	0.0	23.67	2.31	0.0	0.0
original	19.95	0.23	0.0	0.0	0.0	0.0	30.18	0.66	0.0	0.0	348.67	13.58	0.0	0.0
finetune	18.72	0.11	0.0	0.0	1.05	0.61	24.33	0.2	0.0	0.0	223.67	6.66	0.0	0.0
NegGrad	21.74	0.44	4.48	0.34	87.65	2.98	40.05	0.44	21.8	0.66	44.0	5.2	2.33	3.21
CF-k	19.31	0.23	0.0	0.0	0.0	0.0	27.45	0.61	0.0	0.0	294.0	4.36	0.0	0.0
EU-k	17.66	0.23	1.36	0.19	87.9	1.28	16.82	0.79	2.89	0.46	63.67	8.62	91.67	12.58
SCRUB	15.92	0.17	0.2	0.06	87.47	1.46	14.98	0.13	0.39	0.15	54.0	3.61	9.67	2.52

Table 16: Results on CIFAR-10 with All-CNN for the **Resolving Confusion (RC)** application. (Confused class 0,1; 2000-2000 samples).

model	Test error (↓)		Retain error (↓)		Forget error (↑)		IC test error (↓)		IC retain error (↓)		Fgt test error (↓)		Fgt retain error (↓)	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
retrain	2.43	0.32	0.0	0.0	99.83	0.29	3.33	0.58	0.0	0.0	0.0	0.0	0.0	0.0
original	7.37	0.31	1.21	0.12	15.83	4.19	27.67	1.26	8.26	0.8	46.0	4.36	36.33	3.51
finetune	4.17	0.5	0.0	0.0	56.83	9.44	11.17	1.76	0.0	0.0	15.67	3.51	0.0	0.0
NegGrad	5.63	0.38	0.31	0.22	71.33	9.88	19.33	1.04	2.12	1.51	8.33	2.31	0.0	0.0
CF-k	5.4	0.4	0.07	0.06	33.83	3.33	17.33	1.76	0.45	0.39	27.67	3.51	2.0	1.73
EU-k	3.0	0.26	0.0	0.0	90.17	4.65	6.0	1.8	0.0	0.0	2.0	2.65	0.0	0.0
SCRUB	3.07	0.59	0.0	0.0	98.5	0.5	6.83	1.26	0.0	0.0	0.67	0.58	0.0	0.0

Table 17: Results on Lacuna-10 with ResNet for the **Resolving Confusion (RC)** application. (Confused class 0,1; 200-200 samples).

model	Test error (↓)		Retain error (↓)		Forget error (↑)		IC test error (↓)		IC retain error (↓)		Fgt test error (↓)		Fgt retain error (↓)	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
retrain	2.13	0.25	0.0	0.0	99.83	0.29	2.5	1.32	0.0	0.0	0.0	0.0	0.0	0.0
original	7.83	0.55	1.21	0.52	16.0	4.92	31.33	1.76	8.26	3.53	56.33	3.79	36.33	15.53
finetune	3.0	0.7	0.0	0.0	74.5	6.08	6.5	2.0	0.0	0.0	9.0	3.0	0.0	0.0
NegGrad	4.3	0.52	0.4	0.06	89.67	4.25	15.33	2.47	2.73	0.39	4.67	3.21	0.0	0.0
CF-k	5.27	0.47	0.11	0.07	33.33	1.61	18.5	2.29	0.76	0.47	31.33	3.51	3.33	2.08
EU-k	2.53	0.67	0.09	0.02	97.83	2.08	5.17	1.04	0.38	0.35	0.33	0.58	0.67	0.58
SCRUB	2.1	0.4	0.0	0.0	97.5	1.73	4.17	0.58	0.0	0.0	0.33	0.58	0.0	0.0

Table 18: Results on Lacuna-10 with All-CNN for the **Resolving Confusion (RC)** application. (Confused class 0,1; 200-200 samples).

method	Test error mean	Test error std	Forget error mean	Forget error std	Retain error mean	Retain error std	MIA mean	MIA std
Retrain	16.71	0.05	26.67	3.09	0.00	0.00	51.33	6.13
Original	16.71	0.07	0.00	0.00	0.00	0.00	68.67	3.09
Finetune	16.86	0.13	0.00	0.00	0.00	0.00	69.33	2.05
NegGrad	21.65	0.40	47.00	3.74	4.54	0.70	73.00	1.41
CF-k	16.82	0.03	0.00	0.00	0.00	0.00	69.67	1.89
EU-k	18.44	0.21	0.33	0.47	0.32	0.02	66.00	2.94
Bad-T	22.43	0.37	1.67	1.25	10.13	0.15	77.67	4.11
SCRUB	17.01	0.20	33.00	5.89	0.00	0.00	51.00	1.41
SCRUB+R	16.88	0.19	26.33	4.50	0.00	0.00	49.33	2.49

Table 19: Standard MIA for All-CNN architecture on CIFAR-10 for selective unlearning, for the **User Privacy (UP)** application.

method	Test error mean	Test error std	Forget error mean	Forget error std	Retain error mean	Retain error std	MIA mean	MIA std
Retrain	13.98	0.07	100.00	0.00	0.00	0.00	48.73	0.24
Original	15.70	0.09	0.00	0.00	0.00	0.00	71.40	0.70
Finetune	14.53	0.13	1.31	0.54	0.00	0.00	74.97	1.27
NegGrad	17.04	0.11	59.91	1.53	0.43	0.09	70.03	1.92
CF-k	15.72	0.06	0.00	0.00	0.00	0.00	72.93	1.06
EU-k	15.76	0.28	100.00	0.00	0.24	0.02	51.60	1.22
Bad-T	16.98	0.40	81.93	3.50	5.84	0.43	58.07	1.76
SCRUB	14.93	0.17	100.00	0.00	0.09	0.02	54.30	2.24
SCRUB+R	14.93	0.17	100.00	0.00	0.09	0.02	54.30	2.24

Table 20: Standard MIA for All-CNN architecture on CIFAR-10 for class unlearning, for the **User Privacy (UP)** application.

method	Test error mean	Test error std	Forget error mean	Forget error std	Retain error mean	Retain error std	MIA mean	MIA std
Retrain	17.38	0.15	29.33	2.49	0.00	0.00	54.00	1.63
Original	17.41	0.15	0.00	0.00	0.00	0.00	65.33	0.47
Finetune	17.48	0.16	0.00	0.00	0.00	0.00	64.00	0.82
NegGrad	21.69	0.07	45.33	2.62	3.94	0.43	66.67	1.70
CF-k	17.53	0.19	0.00	0.00	0.00	0.00	65.00	0.00
EU-k	19.77	0.04	13.67	0.47	0.06	0.01	53.00	3.27
Bad-T	23.47	1.57	34.67	1.70	14.53	1.65	59.67	4.19
SCRUB	17.01	0.03	71.67	0.94	0.01	0.01	78.00	2.45
SCRUB+R	17.54	0.28	19.33	14.64	0.01	0.01	58.67	1.89

Table 21: Standard MIA for ResNet architecture on CIFAR-10 for selective unlearning, for the **User Privacy (UP)** application.

method	Test error		Forget error		Retain error		MIA	
	mean	std	mean	std	mean	std	mean	std
Retrain	14.69	0.10	100.00	0.00	0.00	0.00	49.33	1.67
Original	16.33	0.14	0.00	0.00	0.00	0.00	71.10	0.67
Finetune	15.10	0.16	0.33	0.17	0.00	0.00	75.57	0.69
NegGrad	17.41	0.09	61.00	1.14	0.44	0.05	69.57	1.19
CF-k	15.29	0.02	0.04	0.04	0.00	0.00	75.73	0.34
EU-k	17.05	0.07	97.48	0.28	0.05	0.01	54.20	2.27
Bad-T	19.56	1.44	11.34	1.82	94.67	6.12	54.33	0.31
SCRUB	15.33	0.06	100.00	0.00	0.08	0.01	52.20	1.71
SCRUB+R	15.33	0.06	100.00	0.00	0.08	0.01	52.20	1.71

Table 22: Standard MIA for ResNet architecture on CIFAR-10 for class unlearning, for the **User Privacy (UP)** application.

method	Test error		Forget error		Retain error		MIA	
	mean	std	mean	std	mean	std	mean	std
Retrain	1.50	0.08	0.33	0.47	0.00	0.00	52.00	2.16
Original	1.57	0.24	0.00	0.00	0.00	0.00	59.00	2.16
Finetune	1.40	0.16	0.00	0.00	0.00	0.00	57.33	3.30
NegGrad	3.60	0.14	14.33	1.25	0.87	0.07	51.00	1.63
CF-k	1.57	0.12	0.00	0.00	0.00	0.00	58.33	2.49
EU-k	3.90	1.47	0.00	0.00	0.76	0.63	52.00	3.56
Bad-T	4.90	2.10	1.34	1.20	0.67	0.94	67.67	6.94
SCRUB	1.67	0.19	0.00	0.00	0.00	0.00	57.67	0.94
SCRUB+R	1.67	0.19	0.00	0.00	0.00	0.00	57.67	0.94

Table 23: Standard MIA for All-CNN architecture on Lacuna-10 for selective unlearning, for the **User Privacy (UP)** application.

method	Test error		Forget error		Retain error		MIA	
	mean	std	mean	std	mean	std	mean	std
Retrain	1.67	0.09	100.00	0.00	0.00	0.00	55.67	2.62
Original	1.70	0.21	0.00	0.00	0.00	0.00	58.00	1.63
Finetune	1.67	0.27	0.00	0.00	0.00	0.00	56.33	1.25
NegGrad	2.00	0.00	14.27	0.74	0.00	0.00	54.33	2.05
CF-k	2.07	0.14	0.00	0.00	0.00	0.00	52.33	2.05
EU-k	4.15	1.22	62.08	44.26	0.81	0.53	52.67	3.68
Bad-T	2.56	0.09	38.65	36.80	0.37	0.18	63.33	2.49
SCRUB	1.96	0.34	100.00	0.00	0.00	0.00	50.33	2.62
SCRUB+R	1.96	0.34	100.00	0.00	0.00	0.00	50.33	2.62

Table 24: Standard MIA for All-CNN architecture on Lacuna-10 for class unlearning, for the **User Privacy (UP)** application.

method	Test error		Forget error		Retain error		MIA	
	mean	std	mean	std	mean	std	mean	std
Retrain	2.50	0.24	1.67	0.94	0.00	0.00	49.67	3.09
Original	2.53	0.25	0.00	0.00	0.00	0.00	56.67	1.70
Finetune	2.67	0.05	0.00	0.00	0.00	0.00	53.67	0.94
NegGrad	4.30	0.43	12.67	3.30	0.95	0.08	54.00	2.16
CF-k	2.47	0.25	0.00	0.00	0.00	0.00	56.00	0.82
EU-k	2.60	0.00	0.00	0.00	0.03	0.00	56.00	2.83
Bad-T	7.30	2.20	3.26	1.83	0.33	0.47	67.33	3.40
SCRUB	2.97	0.25	6.00	3.27	0.00	0.00	50.67	4.03
SCRUB+R	2.97	0.25	6.00	3.27	0.00	0.00	50.67	4.03

Table 25: Standard MIA for ResNet architecture on Lacuna-10 for selective unlearning, for the **User Privacy (UP)** application.

method	Test error		Forget error		Retain error		MIA	
	mean	std	mean	std	mean	std	mean	std
Retrain	2.52	0.19	100.00	0.00	0.00	0.00	55.00	2.94
Original	2.81	0.28	0.00	0.00	0.00	0.00	56.00	2.45
Finetune	3.04	0.19	0.00	0.00	0.00	0.00	54.67	1.25
NegGrad	2.74	0.26	9.48	0.64	0.00	0.00	53.67	4.03
CF-k	2.81	0.28	0.00	0.00	0.00	0.00	56.00	2.45
EU-k	2.48	0.14	7.71	2.52	0.00	0.00	54.33	3.09
Bad-T	3.37	0.50	67.60	24.26	1.06	0.47	58.00	2.94
SCRUB	3.26	0.38	99.90	0.15	0.07	0.05	54.33	2.49
SCRUB+R	3.26	0.38	99.90	0.15	0.07	0.05	54.33	2.49

Table 26: Standard MIA for ResNet architecture on Lacuna-10 for class unlearning, for the **User Privacy (UP)** application.