# Fast Machine Unlearning via Robust Training

**Youssef Allouah** [1 2] **Joshua Kazdan** [1] **Rachid Guerraoui** [2] **Sanmi Koyejo** [1]

## Abstract

Machine unlearning, the process of selectively removing knowledge from trained models, emerges as a crucial mechanism for maintaining model relevance and privacy. However, the effectiveness of unlearning hinges on the quality of training, a challenge exacerbated by sensitivity to outlier data. We introduce the first robust training approach to unlearning tailored to address this challenge by minimizing the training loss on the worst-case retain set to ensure a sturdy initialization for subsequent unlearning. Our method comes with theoretical guarantees for losses satisfying the Polyak-Łojasiewicz inequality, whereas most prior machine unlearning guarantees apply only to convex losses. Through empirical evaluations, we demonstrate the seamless integration of our approach with various unlearning techniques, resulting in accelerated processes and enhanced overall performance.

## 1. Introduction

The ability to selectively remove or "forget" portions of the training data from a deep learning model is a timely and intriguing scientific challenge with profound practical implications. As deep neural networks find widespread deployment across diverse domains like computer vision, natural language processing, speech recognition, and healthcare, there is a growing need to provide individuals with granular control over their personal data. Stringent data protection regulations, such as the European Union's General Data Protection Regulation (Voigt and Von dem Bussche, 2017), enshrine the "right to be forgotten," mandating that individuals can request the erasure of their data from systems that process it. Machine unlearning, the process of systematically removing the influence of specific training examples from a model, has emerged as a promising solution to address this regulatory necessity while preserving the integrity and performance of the system. This research direction has garnered significant attention recently (Cao and Yang, 2015; Guo et al., 2020; Bourtoule et al., 2021; Nguyen et al., 2022), driven by the imperative to develop algorithms that can efficiently and effectively "unlearn" targeted subsets of the training data.

Existing machine unlearning approaches include heuristic algorithms such as fine-tuning on the *retain data*, i.e., the data that is not to be unlearned, via gradient descent along with maximizing the loss on the *forget data*, i.e., the data to be unlearned, via gradient ascent (Graves et al., 2021). There are also more sophisticated techniques that fine-tune only some layers of the model to provoke catastrophic forgetting (Goel et al., 2022). However, these approaches often overlook the training procedure prior to unlearning. Yet, the sensitivity of the training phase to the forget data is central to unlearning. In this direction, there are several techniques with provable unlearning guarantees for convex tasks, reusing noise injection and privacy account tools from differential privacy (Guo et al., 2020; Neel et al., 2021). These guarantees essentially make it hard for a hypothetical adversary to infer the presence of the forget data from the unlearned model, as in membership inference attacks (Shokri et al., 2017). Orthogonal to such a privacy guarantee, it is always desirable to obtain good performance on the retain data upon training, and sometimes bad performance on the forget data, e.g., in when the forget data is chosen because it is outdated or corrupt. In this work focus on the important scenarios where the forget data consists of outlier or corrupt samples, and naively training on all the data makes unlearning hard.

**Contributions.** Our work proposes the first robust training approach for machine unlearning. We show that our training algorithm TRIMGRAD, a simple variant of gradient descent, guarantees a good initialization for subsequent unlearning for non-convex tasks. In particular, we prove that the fine-tuning strategy for unlearning after TRIMGRAD guarantees fast and good performance on the retain data, for non-convex loss functions satisfying the Polyak-Łojasiewicz inequality. Moreover, we provide experiments on vision tasks with neural networks showing that robust training is compatible with several unlearning approaches, and is particularly useful for unlearning outliers or corrupt data.

[1]Stanford University [2]EPFL. Correspondence to: Youssef Allouah <youssef.allouah@epfl.ch>, Sanmi Koyejo <sanmi@cs.stanford.eduw>.

*Work presented at TF2M workshop at ICML 2024*, Vienna, Austria. PMLR 235, 2024. 

## 2. Problem Statement

Consider a training set $\mathcal{S}$ made of $n$ examples $\mathbf{z}_1, \dots, \mathbf{z}_n$ from a data space $\mathcal{Z}$, and a loss function $\ell \colon \mathbb{R}^d \times \mathcal{Z} \to \mathbb{R}$. During the *training* phase, we are first interested in solving the empirical risk minimization problem

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \quad \mathcal{L}(\boldsymbol{\theta}; \mathcal{S}) \coloneqq \frac{1}{|\mathcal{S}|} \sum_{\mathbf{z} \in \mathcal{S}} \ell(\boldsymbol{\theta}; \mathbf{z}). \tag{1}$$

We denote by $\mathcal{A}$ the training procedure on the full set $\mathcal{S}$ aimed at solving (1), and by $\mathcal{A}(\mathcal{S}) \in \mathbb{R}^d$ the model obtained. Moreover, given the trained model $\mathcal{A}(\mathcal{S})$, we are interested in the scenario where a portion of the training set $\mathcal{S}_f \subset \mathcal{S}$ of size $f \coloneqq |\mathcal{S}_f|$, called the *forget set*, is to be removed. That is, we wish to learn only from the samples of the *retain set* defined as $\mathcal{S}_r \coloneqq \mathcal{S} \setminus \mathcal{S}_f$. Ideally, we can retrain using $\mathcal{S}_r$ only, but we can approximate this on the retain data under computational constraints as follows.

**Definition 1** ($(\varepsilon, T)$-approximate retraining)**.** *Let $\mathcal{A}$ be a training procedure taking the dataset $\mathcal{S}$ as input. An iterative unlearning procedure $\mathcal{U}$, taking as inputs $\mathcal{S}_f$ and $\mathcal{A}(\mathcal{S})$, achieves $\varepsilon$-approximate retraining if, in $T$ iterations, it outputs a parameter $\hat{\boldsymbol{\theta}}_T \coloneqq \mathcal{U}(\mathcal{S}_f, \mathcal{A}(\mathcal{S}))$ such that*

$$\mathcal{L}(\hat{\boldsymbol{\theta}}_T; \mathcal{S}_r) - \min_{\boldsymbol{\theta} \in \mathbb{R}^d} \mathcal{L}(\boldsymbol{\theta}; \mathcal{S}_r) \leq \varepsilon. \tag{2}$$

The definition above requires the unlearning procedure to perform similarly to *training from scratch* on the retain set, i.e., disregarding the trained model. In fact, $(\varepsilon, T)$-approximate retraining can be achieved by training from scratch using gradient descent (GD) on the retain set, with $T = \mathcal{O}(\log \frac{\mathcal{L}_0}{\varepsilon})$ iterations for strongly convex tasks (Nesterov et al., 2018), where $\mathcal{L}_0$ is the error due to initialization. Given that we possess a trained model, we are interested in *efficient* approximate retraining solutions, i.e., with a much smaller number of iterations than training from scratch, assuming the computational cost per iteration is alike.

In addition to approximate retraining (Definition 1), which is a performance-centric unlearning objective, several other objectives could be desirable depending on the machine unlearning application. For example, when privacy is a concern, one can additionally require a differential privacy guarantee on the forget data (Guo et al., 2020; Neel et al., 2021; Gupta et al., 2021), and make it hard for an adversary to infer whether the forget data was used in training. Besides, in order to remove toxic data or biases, it is desirable to target bad performance on the forget data (Kurmanji et al., 2024). Nevertheless, it always remains that we at least want to efficiently achieve good performance on the retain data, which is the purpose of Definition 1.

**Sensitivity to outliers.** A natural solution to fast approximate retraining is to fine-tune on the retain data via GD, initialized at the model obtained by training on the full dataset with GD. However, the main obstacle to this method is that the model obtained from training may be too suboptimal on the retain data, especially if the forget data are outliers compared to the full dataset. We formalize this observation below for the quadratic loss.

**Proposition 1.** *Consider the data space $\mathcal{Z} = \mathbb{R}^d$ and the quadratic loss $\ell(\boldsymbol{\theta}; \mathbf{z}) = \|\boldsymbol{\theta} - \mathbf{z}\|^2, \forall \boldsymbol{\theta}, \mathbf{z} \in \mathbb{R}^d$. Let $\boldsymbol{\theta}_\star(\mathcal{S}) \coloneqq \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \mathcal{L}(\boldsymbol{\theta}; \mathcal{S})$ and $\boldsymbol{\theta}_\star(\mathcal{S}_r) \coloneqq \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \mathcal{L}(\boldsymbol{\theta}; \mathcal{S}_r)$ denote the minima of the training loss on the datasets $\mathcal{S}$ and $\mathcal{S}_r$ respectively. Moreover, let $\overline{\mathbf{z}}_{\mathcal{S}_r}$ and $\overline{\mathbf{z}}_{\mathcal{S}_f}$ denote the averages $\frac{1}{|\mathcal{S}_r|} \sum_{\mathbf{z} \in \mathcal{S}_r} \mathbf{z}$ and $\frac{1}{|\mathcal{S}_f|} \sum_{\mathbf{z} \in \mathcal{S}_f} \mathbf{z}$ respectively. We have*

$$\mathcal{L}(\boldsymbol{\theta}_\star(\mathcal{S}); \mathcal{S}_r) - \mathcal{L}(\boldsymbol{\theta}_\star(\mathcal{S}_r); \mathcal{S}_r) = \left(\frac{f}{n}\right)^2 \left\|\overline{\mathbf{z}}_{\mathcal{S}_f} - \overline{\mathbf{z}}_{\mathcal{S}_r}\right\|^2.$$

The observation above shows that, after naive training on the dataset $\mathcal{S}$, the initial suboptimality on the retain data grows with the distance between the average forget data and retain data. If the forget and retain data are significantly different, the trained model may perform poorly on the retain data. That is, the training may have been too sensitive to the forget data. For illustration, we numerically validate this theoretical observation on a least-squares regression task in Figure 5 of the appendix. We use randomly generated synthetic data and track the convergence speed of the loss on the retain set, during unlearning via fine-tuning, after varying the label of a single outlier sample. The initialization can be worse than random, in which case training from scratch is faster for approximate retraining.

## 3. Robust Training with TRIMGRAD

In our approach, prior to unlearning, our goal is to train on the full data without being too sensitive to the forget data. As a consequence, we can expect efficiency improvements during the unlearning phase over naive training. Of course, the difficulty is that we do not know the forget data in advance. Moreover, the latter could consist of outliers, which we know from Proposition 1 makes the task harder. To this end, assuming we know the number (or good upper bound) of forget points $f$, we propose solving a robust optimization problem instead of (1) for training, as follows:

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \max_{\substack{\mathcal{S}_f \subset \mathcal{S} \\ |\mathcal{S}_f| = f}} \quad \mathcal{L}(\boldsymbol{\theta}; \mathcal{S} \setminus \mathcal{S}_f) = \frac{1}{|\mathcal{S} \setminus \mathcal{S}_f|} \sum_{\mathbf{z} \in \mathcal{S} \setminus \mathcal{S}_f} \ell(\boldsymbol{\theta}; \mathbf{z}). \tag{3}$$

Clearly, by obtaining $\varepsilon_0$-approximate global minimizer of (3), we directly satisfy $(\varepsilon_0, 0)$-approximate retraining. Eventually, a subsequent unlearning phase with $T$ iterations, where we know the forget set, allows to achieve $(\varepsilon, T)$-approximate retraining with $\varepsilon \leq \varepsilon_0$.

**Algorithm 1** TRIMGRAD: TRIMMED GRADIENT DESCENT

**Input:** Initial model $\boldsymbol{\theta}_0$, size of forget set $f := |\mathcal{S}_f|$, learning rate $\gamma$, and number of steps $T$.
**for** $t = 0 \dots T-1$ **do**
  Compute the trimmed mean gradient: $\mathbf{r}_t = \mathrm{TM}_f(\nabla\ell(\boldsymbol{\theta}_t; \mathbf{z}_1), \dots, \nabla\ell(\boldsymbol{\theta}_t; \mathbf{z}_n))$
  Update the model: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \gamma \mathbf{r}_t$
**end**

To solve the robust optimization problem in (3), we use a variant of robust gradient descent based on the coordinate-wise trimmed mean of the gradient batch, which we first recall below. Let $\mathbf{g}_1, \dots, \mathbf{g}_n \in \mathbb{R}^d$, we denote by $[\mathbf{g}_i]_k$, the $k$-th coordinate of $\mathbf{g}_i, i \in [n]$. We denote the $i$-th smallest $k$-th coordinate of $\mathbf{g}_1, \dots, \mathbf{g}_n$ by $[\mathbf{g}]_{k:i}$, i.e., $[\mathbf{g}]_{k:1} \leq \dots \leq [\mathbf{g}]_{k:n}$. Then, the coordinate-wise trimmed mean, parametrized by $f < \frac{n}{2}$ and denoted by $\mathrm{TM}_f(\mathbf{g}_1, \dots, \mathbf{g}_n)$, is a vector in $\mathbb{R}^d$ whose $k$-th coordinate is

$$[\mathrm{TM}_f(\mathbf{g}_1, \dots, \mathbf{g}_n)]_k := \frac{1}{n-2f} \sum_{i=f+1}^{n-f} [\mathbf{g}]_{k:i}. \tag{4}$$

TRIMGRAD is an iterative algorithm where at each step, a descent update is made in a robust direction given by the trimmed mean of the gradients, as summarized in Algorithm 1. That is, we update the model as $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \gamma \mathbf{r}_t$, where $\mathbf{r}_t = \mathrm{TM}_f(\nabla\ell(\boldsymbol{\theta}_t; \mathbf{z}_1), \dots, \nabla\ell(\boldsymbol{\theta}_t; \mathbf{z}_n))$. This procedure is inspired by robust distributed optimization methods (Yin et al., 2018; Allouah et al., 2023), where gradients come from potentially misbehaving workers. TRIMGRAD can be followed by an unlearning phase, like fine-tuning on the retain data. Intuitively, the gradients due to outliers in the forget data would be excluded if they are too different compared to the rest. This reduces sensitivity to the forget data at each iteration, without knowing this data. Finally, we observe that TRIMGRAD requires that we unlearn no more than half of the training set at once, which can be expected to hold in practice, and that it is possible to adapt TRIMGRAD for sequential removals (see Appendix D).

## 4. Theoretical Analysis

In this section, we theoretically show that robust training, by solving (3) with TRIMGRAD, produces a good model for subsequent unlearning. Specifically, we show that this model achieves performance on par with training from scratch in a much smaller number of iterations. Henceforth, we simply denote $\mathcal{L}$ instead of $\mathcal{L}(\cdot\,; \mathcal{S}_r)$ when it is clear from the context, unless specified otherwise.

**Standard assumptions.** As is standard in optimization theory, we assume that the loss function is differentiable and $L$-smooth (Nesterov et al., 2018). Moreover, we assume that it satisfies the $\mu$-Polyak-Łojasiewicz (PL) inequality (Karimi et al., 2016), which relaxes strong convexity without requiring convexity. We defer the formal assumptions to Appendix A.1 due to space limitations.

Finally, we make the following standard assumption on the heterogeneity across the retain data, which can be found in previous works in distributed optimization (Karimireddy et al., 2020; Koloskova et al., 2020; Allouah et al., 2024).

**Assumption 1.** *We assume that there exist $\zeta_\star \geq 0, P \geq 1$, such that for all $\boldsymbol{\theta} \in \mathbb{R}^d$*

$$\tfrac{1}{|\mathcal{S}_r|} \textstyle\sum_{\mathbf{z} \in \mathcal{S}_r} \|\nabla\ell(\boldsymbol{\theta}; \mathbf{z})\|^2 \leq \zeta_\star^2 + P \|\nabla\mathcal{L}(\boldsymbol{\theta})\|^2 . \tag{5}$$

Above, the parameter $\zeta_\star$ can be interpreted as a measure of interpolation, i.e., how well we can fit the data (Vaswani et al., 2019). We first analyze the performance of TRIMGRAD, without unlearning, in Theorem 1 below.

**Theorem 1.** *Let assumptions 1-3 hold, and assume that $f = \mathcal{O}\big(\frac{n}{P}\big)$ and $f < \frac{n}{2}$. Then, $T$ iterations of* TRIMGRAD *(Algorithm 1) on the full training set $\mathcal{S}$, without unlearning, satisfy $(\varepsilon, 0)$-approximate retraining, with*

$$\varepsilon \lesssim \frac{f}{n}\zeta_\star^2 + \mathcal{L}_0 \exp\Big(-\frac{\mu}{2L}T\Big), \tag{6}$$

*where $\lesssim$ denotes inequality up to absolute constants and $\mathcal{L}_0 := \mathcal{L}(\boldsymbol{\theta}_0) - \mathcal{L}_\star$.*

Intuitively, the result above shows that robust training produces a model that already performs well on the retain data, so that no unlearning procedure is needed (1) if $\zeta_\star$ is small, i.e., we are close to the interpolation regime on the retain data, and (2) if $\frac{f}{n}$ is small, i.e., there are few samples to be unlearned in total. Beyond these cases, we show that we can achieve fast approximate retraining, for any precision $\varepsilon$, by fine-tuning on the retain set after using TRIMGRAD.

**Theorem 2.** *Let assumptions 1-3 hold, and assume that $f = \mathcal{O}\big(\frac{n}{P}\big)$ and $f < \frac{n}{2}$. Then, for any $\varepsilon > 0$,* TRIMGRAD *(Algorithm 1) on the full training set $\mathcal{S}$, followed by fine-tuning with $T$ iterations of* GD *on the retain set $\mathcal{S}_r$, satisfy $(\varepsilon, T)$-approximate retraining, with*

$$T \lesssim \frac{L}{\mu} \log\left(\frac{f}{n}\frac{\zeta_\star^2}{\mu}\frac{1}{\varepsilon}\right), \tag{7}$$

*where $\lesssim$ denotes inequality up to absolute constants.*

Recall that, under the PL assumption, retraining from scratch can achieve $(\varepsilon, T)$-approximate retraining in $\mathcal{O}\left(\log \frac{\mathcal{L}_0}{\varepsilon}\right)$, where $\mathcal{L}_0$ is the error due to initialization. The result above shows that we can achieve fast approximate retraining if $\frac{f}{n}\zeta_\star^2 \ll \mathcal{L}_0$, i.e., the fraction of the forget data is compared to the error due to initialization, or that we are close to the interpolation regime (Vaswani et al., 2019).

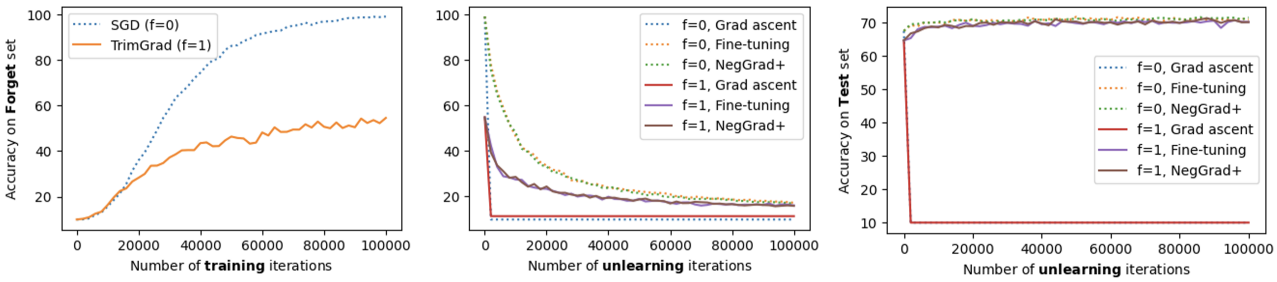


*Figure 1. (Left) Accuracy on the forget set during training*; showing, as expected, that TrimGrad has lower accuracy on the forget set during training. *(Center) Accuracy on the forget set during unleanring*; showing that TrimGrad speeds up unlearning. *(Right) Accuracy on the test set (right) during unlearning*; showing that TrimGrad recovers quickly on the retained set. Methods used for unlearning include gradient ascent on the forget set, fine-tuning on the retain set, and NegGrad+ (Kurmanji et al., 2024). Results for TRIMGRAD are shown with solid lines and results for naive training (SGD) are shown with dotted lines.

Finally, we remark that such a guarantee may not hold for fine-tuning after naive training, as it is independent of the forget data. Indeed, we have shown in Proposition 1 that the model obtained via naive training can behave poorly on the retain data if the forget data are outliers on average.

## 5. Empirical Evaluation

To investigate the effectiveness of robust training for unlearning, we focus on scenarios where the forget set is an outlier compared to the rest of the data on classification tasks. We do so via a simple experiment; we randomly mislabel a fraction of the dataset and compare training algorithms based on how fast we can get a low accuracy on the corrupt data, while keeping a high accuracy on the (clean) test data. In addition to validating the theory in showing fast good performance on the retain data, we also show that TRIMGRAD enables a low forget accuracy on the forget data, unlike naive training which fits all data indiscriminately.

**Experimental setup.** We consider MNIST (LeCun and Cortes, 2005) and CIFAR-10 (Krizhevsky et al., 2014) with fully-connected and convolutional neural networks. All randomness and hyperparameters are fixed across methods. For TRIMGRAD, we apply trimmed mean to micro-batch stochastic gradients for efficiency. Full experimental results and details are in Appendix C.

**Effective unlearning.** In Figure 1, robust training enables rapidly bringing the forget accuracy down to near-baseline levels, much faster than after naive training. Indeed, the same figure shows how the forget accuracy, during training, of TRIMGRAD converges to half of that of naive training. Also, this is the case while the performance on the (clean) test data is on par with the baseline, as shown in Figure 3. It is worth noting that the retain set performance of TRIMGRAD is on par with retraining from scratch, in much lesser iterations (Figure 4), while naive training already fits the retain data, but also the forget data as a downside.

**Efficiency of training and unlearning.** In terms of efficiency, robust training is quite promising. Indeed, the number of unlearning iterations to reach under $20\%$ accuracy on the forget data, i.e., ten points above a random classifier, in about $40,000$ iterations, which is at least 2 times faster than with naive training. Moreover, although the accuracy on the test set (during training) of TRIMGRAD is a few points lower than naive training, the training accuracy converges faster per Figure 4. This is because naive training needs more iterations to perfectly fit the entire training data, but including the forget data. We also observe from Figure 3 that retraining from scratch is substantially slow in achieving good test performance compared to the other methods, as usual in machine unlearning.

## 6. Conclusion

As companies increasingly face lawsuits for training on copyrighted data, the ability to remove copyrighted and harmful materials from models' knowledge bases will become more and more important. We have introduced TRIMGRAD, a novel training algorithm that serves as a primer for subsequent unlearning. By training with TRIMGRAD, one obtains theoretical guarantees of efficient unlearning under mild assumptions. We have provided empirical evidence that applying several unlearning techniques after training with TRIMGRAD yields better rates of unlearning relative than usual training. Future work could apply TRIMGRAD to larger models and more complicated tasks such as image and text generation. This would come with an exploration of how more complicated unlearning methods for language and vision models fare when applied to models pretrained with TRIMGRAD.

## References

Allouah, Y., Farhadkhani, S., Guerraoui, R., Gupta, N., Pinot, R., and Stephan, J. (2023). Fixing by mixing: A

recipe for optimal Byzantine ML under heterogeneity. In *International Conference on Artificial Intelligence and Statistics*, pages 1232–1300. PMLR.

Allouah, Y., Guerraoui, R., Gupta, N., Pinot, R., and Rizk, G. (2024). Robust distributed learning: Tight error bounds and breakdown point under data heterogeneity. *Advances in Neural Information Processing Systems*, 36.

Bourtoule, L., Chandrasekaran, V., Choquette-Choo, C. A., Jia, H., Travers, A., Zhang, B., Lie, D., and Papernot, N. (2021). Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 141–159. IEEE.

Cao, Y. and Yang, J. (2015). Towards making systems forget with machine unlearning. In *2015 IEEE symposium on security and privacy*, pages 463–480. IEEE.

Goel, S., Prabhu, A., Sanyal, A., Lim, S.-N., Torr, P., and Kumaraguru, P. (2022). Towards adversarial evaluations for inexact machine unlearning. *arXiv preprint arXiv:2201.06640*.

Graves, L., Nagisetty, V., and Ganesh, V. (2021). Amnesiac machine learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11516–11524.

Guo, C., Goldstein, T., Hannun, A., and Van Der Maaten, L. (2020). Certified data removal from machine learning models. In *International Conference on Machine Learning*, pages 3832–3842. PMLR.

Gupta, V., Jung, C., Neel, S., Roth, A., Sharifi-Malvajerdi, S., and Waites, C. (2021). Adaptive machine unlearning. *Advances in Neural Information Processing Systems*, 34:16319–16330.

Karimi, H., Nutini, J., and Schmidt, M. (2016). Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part I 16*, pages 795–811. Springer.

Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S., Stich, S., and Suresh, A. T. (2020). Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR.

Koloskova, A., Loizou, N., Boreiri, S., Jaggi, M., and Stich, S. (2020). A unified theory of decentralized SGD with changing topology and local updates. In *International Conference on Machine Learning*, pages 5381–5393. PMLR.

Krizhevsky, A., Nair, V., and Hinton, G. (2014). The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 55(5).

Kurmanji, M., Triantafillou, P., Hayes, J., and Triantafillou, E. (2024). Towards unbounded machine unlearning. *Advances in Neural Information Processing Systems*, 36.

LeCun, Y. and Cortes, C. (2005). The mnist database of handwritten digits.

Neel, S., Roth, A., and Sharifi-Malvajerdi, S. (2021). Descent-to-delete: Gradient-based methods for machine unlearning. In *Algorithmic Learning Theory*, pages 931–962. PMLR.

Nesterov, Y. et al. (2018). *Lectures on convex optimization*, volume 137. Springer.

Nguyen, T. T., Huynh, T. T., Nguyen, P. L., Liew, A. W.-C., Yin, H., and Nguyen, Q. V. H. (2022). A survey of machine unlearning. *arXiv preprint arXiv:2209.02299*.

Sai Abhishek, A. V. (2022). Resnet18 model with sequential layer for computing accuracy on image classification dataset. 10:2320–2882.

Shokri, R., Stronati, M., Song, C., and Shmatikov, V. (2017). Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE.

Vaswani, S., Bach, F., and Schmidt, M. (2019). Fast and faster convergence of SGD for over-parameterized models and an accelerated perceptron. In *The 22nd international conference on artificial intelligence and statistics*, pages 1195–1204. PMLR.

Voigt, P. and Von dem Bussche, A. (2017). The EU general data protection regulation (GDPR). *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 10(3152676):10–5555.

Yin, D., Chen, Y., Kannan, R., and Bartlett, P. (2018). Byzantine-robust distributed learning: Towards optimal statistical rates. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5650–5659. PMLR.

# A. Assumptions and Proofs

## A.1. Assumptions

**Assumption 2** ($L$-smoothness)**.** *A function $\mathcal{L}\colon \mathbb{R}^d \to \mathbb{R}$ is $L$-smooth if, for all $\boldsymbol{\theta}, \boldsymbol{\theta}' \in \mathbb{R}^d$, we have*

$$\mathcal{L}(\boldsymbol{\theta}') - \mathcal{L}(\boldsymbol{\theta}) - \langle \nabla \mathcal{L}(\boldsymbol{\theta}), \boldsymbol{\theta}' - \boldsymbol{\theta} \rangle \leq \frac{L}{2} \left\| \boldsymbol{\theta}' - \boldsymbol{\theta} \right\|^2.$$

The above is equivalent to, for all $\boldsymbol{\theta}, \boldsymbol{\theta}' \in \mathbb{R}^d$, having $\|\nabla \mathcal{L}(\boldsymbol{\theta}') - \nabla \mathcal{L}(\boldsymbol{\theta})\| \leq L \|\boldsymbol{\theta}' - \boldsymbol{\theta}\|$ (see, e.g., (Nesterov et al., 2018)). Moreover, some of our results will be derived assuming the Polyak-Łojasiewicz (PL) inequality (Karimi et al., 2016) shown below, which relaxes strong convexity without requiring convexity.

**Assumption 3** ($\mu$-Polyak-Łojasiewicz (PL))**.** *A function $\mathcal{L}\colon \mathbb{R}^d \to \mathbb{R}$ is $\mu$-PL if, for all $\boldsymbol{\theta} \in \mathbb{R}^d$, we have*

$$2\mu \left( \mathcal{L}(\boldsymbol{\theta}) - \mathcal{L}_\star \right) \leq \left\| \nabla \mathcal{L}(\boldsymbol{\theta}) \right\|^2,$$

*where $\mathcal{L}_* \coloneqq \inf_{\boldsymbol{\theta} \in \mathbb{R}^d} \mathcal{L}(\boldsymbol{\theta})$.*

Note that a function satisfies $L$-smoothness and $\mu$-PL inequality simultaneously only if $\mu \leq L$.

## A.2. Proofs

**Proposition 1.** *Consider the data space $\mathcal{Z} = \mathbb{R}^d$ and the quadratic loss $\ell(\boldsymbol{\theta}; \mathbf{z}) = \|\boldsymbol{\theta} - \mathbf{z}\|^2, \forall \boldsymbol{\theta}, \mathbf{z} \in \mathbb{R}^d$. Let $\boldsymbol{\theta}_\star(\mathcal{S}) \coloneqq \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \mathcal{L}(\boldsymbol{\theta}; \mathcal{S})$ and $\boldsymbol{\theta}_\star(\mathcal{S}_r) \coloneqq \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \mathcal{L}(\boldsymbol{\theta}; \mathcal{S}_r)$ denote the minima of the training loss on the datasets $\mathcal{S}$ and $\mathcal{S}_r$ respectively. Moreover, let $\overline{\mathbf{z}}_{\mathcal{S}_r}$ and $\overline{\mathbf{z}}_{\mathcal{S}_f}$ denote the averages $\frac{1}{|\mathcal{S}_r|} \sum_{\mathbf{z} \in \mathcal{S}_r} \mathbf{z}$ and $\frac{1}{|\mathcal{S}_f|} \sum_{\mathbf{z} \in \mathcal{S}_f} \mathbf{z}$ respectively. We have*

$$\mathcal{L}(\boldsymbol{\theta}_\star(\mathcal{S}); \mathcal{S}_r) - \mathcal{L}(\boldsymbol{\theta}_\star(\mathcal{S}_r); \mathcal{S}_r) = \left( \frac{f}{n} \right)^2 \left\| \overline{\mathbf{z}}_{\mathcal{S}_f} - \overline{\mathbf{z}}_{\mathcal{S}_r} \right\|^2.$$

*Proof.* Consider the data space $\mathcal{Z} = \mathbb{R}^d$ and the quadratic loss $\ell(\boldsymbol{\theta}; \mathbf{z}) = \|\boldsymbol{\theta} - \mathbf{z}\|^2, \forall \boldsymbol{\theta}, \mathbf{z} \in \mathbb{R}^d$. Let $\overline{\mathbf{z}}_{\mathcal{S}}$, $\overline{\mathbf{z}}_{\mathcal{S}_r}$ and $\overline{\mathbf{z}}_{\mathcal{S}_f}$ denote the averages $\frac{1}{|\mathcal{S}|} \sum_{\mathbf{z} \in \mathcal{S}} \mathbf{z}$, $\frac{1}{|\mathcal{S}_r|} \sum_{\mathbf{z} \in \mathcal{S}_r} \mathbf{z}$ and $\frac{1}{|\mathcal{S}_r|} \sum_{\mathbf{z} \in \mathcal{S}_f} \mathbf{z}$ respectively.

First, observe that

$$\boldsymbol{\theta}_\star(\mathcal{S}) = \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \left\{ \mathcal{L}(\boldsymbol{\theta}; \mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{\mathbf{z} \in \mathcal{S}} \|\boldsymbol{\theta} - \mathbf{z}\|^2 \right\} = \overline{\mathbf{z}}_{\mathcal{S}},$$

and similarly $\boldsymbol{\theta}_\star(\mathcal{S}_r) = \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \mathcal{L}(\boldsymbol{\theta}; \mathcal{S}_r) = \overline{\mathbf{z}}_{\mathcal{S}_r}$.

On the one hand, using bias-variance decomposition and recalling that $\mathcal{S}_r = \mathcal{S} \setminus \mathcal{S}_f$, we have

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}_\star(\mathcal{S}); \mathcal{S}_r) = \mathcal{L}(\overline{\mathbf{z}}_{\mathcal{S}}; \mathcal{S}_r) &= \frac{1}{|\mathcal{S}_r|} \sum_{\mathbf{z} \in \mathcal{S}_r} \|\overline{\mathbf{z}}_{\mathcal{S}} - \mathbf{z}\|^2 = \frac{1}{|\mathcal{S}_r|} \left[ \sum_{\mathbf{z} \in \mathcal{S}} \|\overline{\mathbf{z}}_{\mathcal{S}} - \mathbf{z}\|^2 - \sum_{\mathbf{z} \in \mathcal{S}_f} \|\overline{\mathbf{z}}_{\mathcal{S}} - \mathbf{z}\|^2 \right] \\
&= \frac{1}{|\mathcal{S}_r|} \left[ \sum_{\mathbf{z} \in \mathcal{S}} \|\overline{\mathbf{z}}_{\mathcal{S}} - \mathbf{z}\|^2 - \left( \sum_{\mathbf{z} \in \mathcal{S}_f} \left\|\overline{\mathbf{z}}_{\mathcal{S}_f} - \mathbf{z}\right\|^2 + |\mathcal{S}_f| \left\|\overline{\mathbf{z}}_{\mathcal{S}_f} - \overline{\mathbf{z}}_{\mathcal{S}}\right\|^2 \right) \right] \\
&= \frac{1}{|\mathcal{S}_r|} \left[ \sum_{\mathbf{z} \in \mathcal{S}} \|\overline{\mathbf{z}}_{\mathcal{S}} - \mathbf{z}\|^2 - \sum_{\mathbf{z} \in \mathcal{S}_f} \left\|\overline{\mathbf{z}}_{\mathcal{S}_f} - \mathbf{z}\right\|^2 \right] - \frac{|\mathcal{S}_f|}{|\mathcal{S}_r|} \left\|\overline{\mathbf{z}}_{\mathcal{S}_f} - \overline{\mathbf{z}}_{\mathcal{S}}\right\|^2.
\end{aligned}$$

On the other hand, we similarly have

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}_\star(\mathcal{S}_r);\mathcal{S}_r) = \mathcal{L}(\overline{\mathbf{z}}_{\mathcal{S}_r};\mathcal{S}_r) &= \frac{1}{|\mathcal{S}_r|}\sum_{\mathbf{z}\in\mathcal{S}_r}\left\|\overline{\mathbf{z}}_{\mathcal{S}_r}-\mathbf{z}\right\|^2 = \frac{1}{|\mathcal{S}_r|}\left[\sum_{\mathbf{z}\in\mathcal{S}}\left\|\overline{\mathbf{z}}_{\mathcal{S}_r}-\mathbf{z}\right\|^2 - \sum_{\mathbf{z}\in\mathcal{S}_f}\left\|\overline{\mathbf{z}}_{\mathcal{S}_r}-\mathbf{z}\right\|^2\right] \\ &= \frac{1}{|\mathcal{S}_r|}\left[\left(\sum_{\mathbf{z}\in\mathcal{S}}\left\|\overline{\mathbf{z}}_{\mathcal{S}}-\mathbf{z}\right\|^2 + |\mathcal{S}|\left\|\overline{\mathbf{z}}_{\mathcal{S}_r}-\overline{\mathbf{z}}_{\mathcal{S}}\right\|^2\right) - \left(\sum_{\mathbf{z}\in\mathcal{S}_f}\left\|\overline{\mathbf{z}}_{\mathcal{S}_f}-\mathbf{z}\right\|^2 + |\mathcal{S}_f|\left\|\overline{\mathbf{z}}_{\mathcal{S}_f}-\overline{\mathbf{z}}_{\mathcal{S}_r}\right\|^2\right)\right] \\ &= \frac{1}{|\mathcal{S}_r|}\left[\sum_{\mathbf{z}\in\mathcal{S}}\left\|\overline{\mathbf{z}}_{\mathcal{S}}-\mathbf{z}\right\|^2 - \sum_{\mathbf{z}\in\mathcal{S}_f}\left\|\overline{\mathbf{z}}_{\mathcal{S}_f}-\mathbf{z}\right\|^2\right] + \frac{|\mathcal{S}|\left\|\overline{\mathbf{z}}_{\mathcal{S}_r}-\overline{\mathbf{z}}_{\mathcal{S}}\right\|^2 - |\mathcal{S}_f|\left\|\overline{\mathbf{z}}_{\mathcal{S}_r}-\overline{\mathbf{z}}_{\mathcal{S}_f}\right\|^2}{|\mathcal{S}_r|}.\end{aligned}$$

By combining the above two equations, we obtain

$$\mathcal{L}(\boldsymbol{\theta}_\star(\mathcal{S});\mathcal{S}_r) - \mathcal{L}(\boldsymbol{\theta}_\star(\mathcal{S}_r);\mathcal{S}_r) = \frac{|\mathcal{S}_f|\left\|\overline{\mathbf{z}}_{\mathcal{S}_r}-\overline{\mathbf{z}}_{\mathcal{S}_f}\right\|^2 - |\mathcal{S}|\left\|\overline{\mathbf{z}}_{\mathcal{S}_r}-\overline{\mathbf{z}}_{\mathcal{S}}\right\|^2 - |\mathcal{S}_f|\left\|\overline{\mathbf{z}}_{\mathcal{S}_f}-\overline{\mathbf{z}}_{\mathcal{S}}\right\|^2}{|\mathcal{S}_r|}.$$

Now, observe that $\overline{\mathbf{z}}_{\mathcal{S}} = \frac{1}{|\mathcal{S}|}\sum_{\mathbf{z}\in\mathcal{S}}\mathbf{z} = \frac{|\mathcal{S}_r|}{|\mathcal{S}|}\overline{\mathbf{z}}_{\mathcal{S}_r} + \frac{|\mathcal{S}_f|}{|\mathcal{S}|}\overline{\mathbf{z}}_{\mathcal{S}_f}$, so that $\overline{\mathbf{z}}_{\mathcal{S}} - \overline{\mathbf{z}}_{\mathcal{S}_r} = \frac{|\mathcal{S}_f|}{|\mathcal{S}|}(\overline{\mathbf{z}}_{\mathcal{S}_f} - \overline{\mathbf{z}}_{\mathcal{S}_r})$ and $\overline{\mathbf{z}}_{\mathcal{S}} - \overline{\mathbf{z}}_{\mathcal{S}_f} = \frac{|\mathcal{S}_r|}{|\mathcal{S}|}(\overline{\mathbf{z}}_{\mathcal{S}_r} - \overline{\mathbf{z}}_{\mathcal{S}_f})$, because $|\mathcal{S}| = |\mathcal{S}_f| + |\mathcal{S}_r|$. Plugging these identities in the equation above then yields

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}_\star(\mathcal{S});\mathcal{S}_r) - \mathcal{L}(\boldsymbol{\theta}_\star(\mathcal{S}_r);\mathcal{S}_r) &= \frac{|\mathcal{S}_f|\left\|\overline{\mathbf{z}}_{\mathcal{S}_r}-\overline{\mathbf{z}}_{\mathcal{S}_f}\right\|^2 - \frac{|\mathcal{S}_f|^2}{|\mathcal{S}|}\left\|\overline{\mathbf{z}}_{\mathcal{S}_r}-\overline{\mathbf{z}}_{\mathcal{S}_f}\right\|^2 - \frac{|\mathcal{S}_r|^2|\mathcal{S}_f|}{|\mathcal{S}|^2}\left\|\overline{\mathbf{z}}_{\mathcal{S}_f}-\overline{\mathbf{z}}_{\mathcal{S}_r}\right\|^2}{|\mathcal{S}_r|} \\ &= \left(\frac{|\mathcal{S}_f|}{|\mathcal{S}_r|} - \frac{|\mathcal{S}_f|^2}{|\mathcal{S}_r|\,|\mathcal{S}|} - \frac{|\mathcal{S}_r|\,|\mathcal{S}_f|}{|\mathcal{S}|^2}\right)\left\|\overline{\mathbf{z}}_{\mathcal{S}_f}-\overline{\mathbf{z}}_{\mathcal{S}_r}\right\|^2.\end{aligned}$$

Recall the notation $f = |\mathcal{S}_f|$ and $n = |\mathcal{S}|$, so that $|\mathcal{S}_r| = n - f$ and

$$\mathcal{L}(\boldsymbol{\theta}_\star(\mathcal{S});\mathcal{S}_r) - \mathcal{L}(\boldsymbol{\theta}_\star(\mathcal{S}_r);\mathcal{S}_r) = \left(\frac{f}{n-f} - \frac{f^2}{n(n-f)} - \frac{f(n-f)}{n^2}\right)\left\|\overline{\mathbf{z}}_{\mathcal{S}_f}-\overline{\mathbf{z}}_{\mathcal{S}_r}\right\|^2 = \left(\frac{f}{n}\right)^2\left\|\overline{\mathbf{z}}_{\mathcal{S}_f}-\overline{\mathbf{z}}_{\mathcal{S}_r}\right\|^2.$$

The above concludes the proof. □

**Lemma 3** (Proposition 2, (Allouah et al., 2023), paraphrased)**.** *Let $n \in \mathbb{N}^*$ and $f < n/2$. For any $\mathbf{g}_1, \ldots, \mathbf{g}_n \in \mathbb{R}^d$ and any $\mathcal{I} \subseteq [n]$ of size $|\mathcal{I}| = n - f$, we have*

$$\left\|\mathrm{TM}_f(\mathbf{g}_1,\ldots,\mathbf{g}_n) - \overline{\mathbf{g}}_{\mathcal{I}}\right\|^2 \leq \frac{6f}{n-2f}\left(1 + \frac{f}{n-2f}\right)\frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}}\left\|\mathbf{g}_i - \overline{\mathbf{g}}_{\mathcal{I}}\right\|^2, \tag{8}$$

*where we denote the average $\overline{\mathbf{g}}_{\mathcal{I}} \coloneqq \frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}}\mathbf{g}_i$.*

**Theorem 1.** *Let assumptions 1-3 hold, and assume that $f = \mathcal{O}\left(\frac{n}{P}\right)$ and $f < \frac{n}{2}$. Then, $T$ iterations of* TrimGrad *(Algorithm 1) on the full training set $\mathcal{S}$, without unlearning, satisfy $(\varepsilon, 0)$-approximate retraining, with*

$$\varepsilon \lesssim \frac{f}{n}\zeta_\star^2 + \mathcal{L}_0\exp\left(-\frac{\mu}{2L}T\right), \tag{6}$$

*where $\lesssim$ denotes inequality up to absolute constants and $\mathcal{L}_0 \coloneqq \mathcal{L}(\boldsymbol{\theta}_0) - \mathcal{L}_\star$.*

*Proof.* Let $t \in \{0, \ldots, T-1\}$. Recall that $\mathcal{L}$ is $L$-smooth by assumption. From Algorithm 1, recall that $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \gamma\mathbf{r}_t$. Hence, by the smoothness assumption, we have

$$\mathcal{L}(\boldsymbol{\theta}_{t+1}) - \mathcal{L}(\boldsymbol{\theta}_t) \leq -\gamma\left\langle\nabla\mathcal{L}(\boldsymbol{\theta}_t), \mathbf{r}_t\right\rangle + \frac{1}{2}\gamma^2 L\left\|\mathbf{r}_t\right\|^2. \tag{9}$$

Moreover, we recall the identity

$$\langle \nabla \mathcal{L}(\boldsymbol{\theta}_t), \mathbf{r}_t \rangle = \frac{1}{2}\left( \|\nabla \mathcal{L}(\boldsymbol{\theta}_t)\|^2 + \|\mathbf{r}_t\|^2 - \|\nabla \mathcal{L}(\boldsymbol{\theta}_t) - \mathbf{r}_t\|^2 \right).$$

Substituting the above in (18) we obtain that

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}_{t+1}) - \mathcal{L}(\boldsymbol{\theta}_t) &\leq -\frac{\gamma}{2}\left( \|\nabla \mathcal{L}(\boldsymbol{\theta}_t)\|^2 + \|\mathbf{r}_t\|^2 - \|\nabla \mathcal{L}(\boldsymbol{\theta}_t) - \mathbf{r}_t\|^2 \right) + \frac{1}{2}\gamma^2 L \|\mathbf{r}_t\|^2 \\ &= -\frac{\gamma}{2}\|\nabla \mathcal{L}(\boldsymbol{\theta}_t)\|^2 - \frac{\gamma}{2}(1-\gamma L)\|\mathbf{r}_t\|^2 + \frac{\gamma}{2}\|\nabla \mathcal{L}(\boldsymbol{\theta}_t) - \mathbf{r}_t\|^2 .\end{aligned}$$

Substituting $\gamma = \frac{1}{L}$ in the above we obtain that

$$\mathcal{L}(\boldsymbol{\theta}_{t+1}) - \mathcal{L}(\boldsymbol{\theta}_t) \leq -\frac{1}{2L}\|\nabla \mathcal{L}(\boldsymbol{\theta}_t)\|^2 + \frac{1}{2L}\|\mathbf{r}_t - \nabla \mathcal{L}(\boldsymbol{\theta}_t)\|^2 . \tag{10}$$

By applying Lemma 3 to the vectors $\nabla \ell(\boldsymbol{\theta}_t; \mathbf{z}_1), \ldots, \nabla \ell(\boldsymbol{\theta}_t; \mathbf{z}_n)$ and the indices set $\mathcal{I} \coloneqq \{i \in [n] \colon \mathbf{z}_i \in \mathcal{S}_r\}$, and denoting $\kappa \coloneqq \frac{6f}{n-2f}\left(1 + \frac{f}{n-2f}\right)$, we obtain

$$\begin{aligned}\|\mathbf{r}_t - \nabla \mathcal{L}(\boldsymbol{\theta}_t)\|^2 &= \left\| \mathrm{TM}_f(\nabla \ell(\boldsymbol{\theta}_t; \mathbf{z}_1), \ldots, \nabla \ell(\boldsymbol{\theta}_t; \mathbf{z}_n)) - \frac{1}{|\mathcal{I}|}\sum_{j \in \mathcal{I}} \nabla \ell(\boldsymbol{\theta}_t; \mathbf{z}_j) \right\|^2 \\ &\leq \frac{\kappa}{|\mathcal{I}|}\sum_{i \in \mathcal{I}} \left\| \nabla \ell(\boldsymbol{\theta}_t; \mathbf{z}_i) - \frac{1}{|\mathcal{I}|}\sum_{j \in \mathcal{I}} \nabla \ell(\boldsymbol{\theta}_t; \mathbf{z}_j) \right\|^2 = \frac{\kappa}{|\mathcal{I}|}\sum_{i \in \mathcal{I}} \|\nabla \ell(\boldsymbol{\theta}_t; \mathbf{z}_i) - \nabla \mathcal{L}(\boldsymbol{\theta}_t)\|^2 .\end{aligned} \tag{11}$$

Besides, Assumption 1 implies that we have

$$\frac{1}{|\mathcal{I}|}\sum_{i \in \mathcal{I}} \|\nabla \ell(\boldsymbol{\theta}_t; \mathbf{z}_i) - \nabla \mathcal{L}(\boldsymbol{\theta}_t)\|^2 = \frac{1}{|\mathcal{I}|}\sum_{i \in \mathcal{I}} \|\nabla \ell(\boldsymbol{\theta}_t; \mathbf{z}_i)\|^2 - \|\nabla \mathcal{L}(\boldsymbol{\theta}_t)\|^2 \leq \zeta_\star^2 + (P-1)\|\nabla \mathcal{L}(\boldsymbol{\theta}_t)\|^2 .$$

Using the above in (11) yields

$$\|\mathbf{r}_t - \nabla \mathcal{L}(\boldsymbol{\theta}_t)\|^2 \leq \kappa\zeta_\star^2 + \kappa(P-1)\|\nabla \mathcal{L}(\boldsymbol{\theta}_t)\|^2 .$$

Substituting the above in (10) yields

$$\mathcal{L}(\boldsymbol{\theta}_{t+1}) - \mathcal{L}(\boldsymbol{\theta}_t) \leq -\frac{1}{2L}\|\nabla \mathcal{L}(\boldsymbol{\theta}_t)\|^2 + \frac{1}{2L}\left( \kappa\zeta_\star^2 + \kappa(P-1)\|\nabla \mathcal{L}(\boldsymbol{\theta}_t)\|^2 \right). \tag{12}$$

Multiplying both sides in (12) by $2L$ and rearranging terms, we get

$$(1 - \kappa(P-1))\|\nabla \mathcal{L}(\boldsymbol{\theta}_t)\|^2 \leq \kappa\zeta_\star^2 + 2L\left(\mathcal{L}(\boldsymbol{\theta}_t) - \mathcal{L}(\boldsymbol{\theta}_{t+1})\right). \tag{13}$$

Averaging over all $t \in \{0, \ldots, T-1\}$ and recalling that $\mathcal{L}_\star = \inf_{\boldsymbol{\theta} \in \mathbb{R}^d} \mathcal{L}(\boldsymbol{\theta})$ yields

$$\begin{aligned}(1 - \kappa(P-1))\frac{1}{T}\sum_{t=0}^{T-1}\|\nabla \mathcal{L}(\boldsymbol{\theta}_t)\|^2 &\leq \kappa\zeta_\star^2 + \frac{2L}{T}\sum_{t=0}^{T-1}\left(\mathcal{L}(\boldsymbol{\theta}_t) - \mathcal{L}(\boldsymbol{\theta}_{t+1})\right) = \kappa\zeta_\star^2 + \frac{2L}{T}\left(\mathcal{L}(\boldsymbol{\theta}_0) - \mathcal{L}(\boldsymbol{\theta}_T)\right) \\ &\leq \kappa\zeta_\star^2 + \frac{2L}{T}\left(\mathcal{L}(\boldsymbol{\theta}_0) - \mathcal{L}_\star\right).\end{aligned}$$

Finally, since we assume that $1 - \kappa(P-1) > 0$, dividing both sides in the above by $1 - \kappa(P-1)$ yields

$$\frac{1}{T}\sum_{t=0}^{T-1}\|\nabla \mathcal{L}(\boldsymbol{\theta}_t)\|^2 \leq \frac{\kappa\zeta_\star^2}{1 - \kappa(P-1)} + \frac{2L\left(\mathcal{L}(\boldsymbol{\theta}_0) - \mathcal{L}_\star\right)}{(1 - \kappa(P-1))T}.$$

Then, outputting the iterate $\boldsymbol{\theta}_\tau$ such that $\tau \in \arg\min_{t \in \{0, \ldots, T-1\}} \|\nabla \mathcal{L}(\boldsymbol{\theta}_t)\|$ and remarking that $\frac{\kappa}{1-\kappa(P-1)} \leq 45\frac{f}{n}$ and $1 - \kappa(P-1) \geq \frac{1}{2}$ when $\frac{f}{n} \leq \min\left\{\frac{1}{3}, \frac{12}{5(P-1)}\right\}$ concludes the proof for the non-convex case.

**PL case.** Assume now that $\mathcal{L}$ is $\mu$-PL. We reuse (13) as follows

$$\begin{aligned}(1-\kappa(P-1))\left\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\right\|^2 &\leq \kappa\zeta_\star^2 + 2L\left(\mathcal{L}(\boldsymbol{\theta}_t) - \mathcal{L}(\boldsymbol{\theta}_{t+1})\right)\\ &= \kappa\zeta_\star^2 + 2L\left(\mathcal{L}(\boldsymbol{\theta}_t) - \mathcal{L}_\star + \mathcal{L}_\star - \mathcal{L}(\boldsymbol{\theta}_{t+1})\right).\end{aligned}$$

Rearranging terms, we get

$$2L\left(\mathcal{L}(\boldsymbol{\theta}_{t+1}) - \mathcal{L}_\star\right) \leq \kappa\zeta_\star^2 - (1-\kappa(P-1))\left\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\right\|^2 + 2L\left(\mathcal{L}(\boldsymbol{\theta}_t) - \mathcal{L}_\star\right).$$

Since $\mathcal{L}$ is $\mu$-PL, we obtain

$$\begin{aligned}2L\left(\mathcal{L}(\boldsymbol{\theta}_{t+1}) - \mathcal{L}_\star\right) &\leq \kappa\zeta_\star^2 - 2\mu\left(1-\kappa(P-1)\right)\left(\mathcal{L}(\boldsymbol{\theta}_t) - \mathcal{L}_\star\right) + 2L\left(\mathcal{L}(\boldsymbol{\theta}_t) - \mathcal{L}_\star\right)\\ &= \kappa\zeta_\star^2 + \left(2L - 2\mu\left(1-\kappa(P-1)\right)\right)\left(\mathcal{L}(\boldsymbol{\theta}_t) - \mathcal{L}_\star\right).\end{aligned}$$

Dividing both sides by $2L$, we get

$$\mathcal{L}(\boldsymbol{\theta}_{t+1}) - \mathcal{L}_\star \leq \frac{\kappa\zeta_\star^2}{2L} + \left(1 - \frac{\mu}{L}\left(1-\kappa(P-1)\right)\right)\left(\mathcal{L}(\boldsymbol{\theta}_t) - \mathcal{L}_\star\right). \tag{14}$$

Then, applying (16) recursively for time indices in $k \in \{0, \ldots, t-1\}$ yields

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}_{t+1}) - \mathcal{L}_\star &\leq \frac{\kappa\zeta_\star^2}{2L}\sum_{k=0}^{t}\left(1 - \frac{\mu}{L}\left(1-\kappa(P-1)\right)\right)^k\\ &\quad + \left(1 - \frac{\mu}{L}\left(1-\kappa(P-1)\right)\right)^{t+1}\left(\mathcal{L}(\boldsymbol{\theta}_0) - \mathcal{L}_\star\right)\\ &\leq \frac{\kappa\zeta_\star^2}{2L}\frac{1}{1-\left(1-\frac{\mu}{L}\left(1-\kappa(P-1)\right)\right)} + \left(1 - \frac{\mu}{L}\left(1-\kappa(P-1)\right)\right)^{t+1}\left(\mathcal{L}(\boldsymbol{\theta}_0) - \mathcal{L}_\star\right)\\ &= \frac{\kappa\zeta_\star^2}{2\mu\left(1-\kappa(P-1)\right)} + \left(1 - \frac{\mu}{L}\left(1-\kappa(P-1)\right)\right)^{t+1}\left(\mathcal{L}(\boldsymbol{\theta}_0) - \mathcal{L}_\star\right).\end{aligned}$$

Using the fact that $(1+x)^n \leq e^{nx}$ for all $x \in \mathbb{R}$ and substituting $t = T-1$ yields

$$\mathcal{L}(\boldsymbol{\theta}_T) - \mathcal{L}_\star \leq \frac{\kappa\zeta_\star^2}{2\mu\left(1-\kappa(P-1)\right)} + e^{-\frac{\mu}{L}\left(1-\kappa(P-1)\right)T}\left(\mathcal{L}(\boldsymbol{\theta}_0) - \mathcal{L}_\star\right).$$

Then, remarking that $\frac{\kappa}{1-\kappa(P-1)} \leq 45\frac{f}{n}$ and $1-\kappa(P-1) \geq \frac{1}{2}$ when $\frac{f}{n} \leq \min\left\{\frac{1}{3}, \frac{12}{5(P-1)}\right\}$ yields that

$$\mathcal{L}(\boldsymbol{\theta}_T) - \mathcal{L}_\star \leq \frac{45}{2\mu}\frac{f}{n}\zeta_\star^2 + e^{-\frac{\mu}{2L}T}\left(\mathcal{L}(\boldsymbol{\theta}_0) - \mathcal{L}_\star\right). \tag{15}$$

The above concludes the proof. □

**Theorem 2.** *Let assumptions 1-3 hold, and assume that $f = \mathcal{O}\left(\frac{n}{P}\right)$ and $f < \frac{n}{2}$. Then, for any $\varepsilon > 0$,* TRIMGRAD *(Algorithm 1) on the full training set $\mathcal{S}$, followed by fine-tuning with $T$ iterations of* GD *on the retain set $\mathcal{S}_r$, satisfy $(\varepsilon, T)$-approximate retraining, with*

$$T \lesssim \frac{L}{\mu}\log\left(\frac{f}{n}\frac{\zeta_\star^2}{\mu}\frac{1}{\varepsilon}\right), \tag{7}$$

*where $\lesssim$ denotes inequality up to absolute constants.*

*Proof.* Assume now that $\mathcal{L}$ is $\mu$-PL. Also, for clarity denote by $T_1$ the number of iterations of TGD prior to the $T$ fine-tuning iterations of gradient descent. Following the proof of Theorem 1 up until (13) yields

$$\begin{aligned}(1-\kappa(P-1))\left\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\right\|^2 &\leq \kappa\zeta_\star^2 + 2L\left(\mathcal{L}(\boldsymbol{\theta}_t) - \mathcal{L}(\boldsymbol{\theta}_{t+1})\right)\\ &= \kappa\zeta_\star^2 + 2L\left(\mathcal{L}(\boldsymbol{\theta}_t) - \mathcal{L}_\star + \mathcal{L}_\star - \mathcal{L}(\boldsymbol{\theta}_{t+1})\right).\end{aligned}$$

Rearranging terms, we get

$$2L\left(\mathcal{L}(\boldsymbol{\theta}_{t+1})-\mathcal{L}_\star\right)\leq\kappa\zeta_\star^2-\left(1-\kappa(P-1)\right)\left\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\right\|^2+2L\left(\mathcal{L}(\boldsymbol{\theta}_t)-\mathcal{L}_\star\right).$$

Since $\mathcal{L}$ is $\mu$-PL, we obtain

$$\begin{aligned}2L\left(\mathcal{L}(\boldsymbol{\theta}_{t+1})-\mathcal{L}_\star\right)&\leq\kappa\zeta_\star^2-2\mu\left(1-\kappa(P-1)\right)\left(\mathcal{L}(\boldsymbol{\theta}_t)-\mathcal{L}_\star\right)+2L\left(\mathcal{L}(\boldsymbol{\theta}_t)-\mathcal{L}_\star\right)\\&=\kappa\zeta_\star^2+\left(2L-2\mu\left(1-\kappa(P-1)\right)\right)\left(\mathcal{L}(\boldsymbol{\theta}_t)-\mathcal{L}_\star\right).\end{aligned}$$

Dividing both sides by $2L$, we get

$$\mathcal{L}(\boldsymbol{\theta}_{t+1})-\mathcal{L}_\star\leq\frac{\kappa\zeta_\star^2}{2L}+\left(1-\frac{\mu}{L}\left(1-\kappa(P-1)\right)\right)\left(\mathcal{L}(\boldsymbol{\theta}_t)-\mathcal{L}_\star\right).\tag{16}$$

Then, applying (16) recursively for time indices in $k\in\{0,\ldots,t-1\}$ yields

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}_{t+1})-\mathcal{L}_\star&\leq\frac{\kappa\zeta_\star^2}{2L}\sum_{k=0}^{t}\left(1-\frac{\mu}{L}\left(1-\kappa(P-1)\right)\right)^k\\&\quad+\left(1-\frac{\mu}{L}\left(1-\kappa(P-1)\right)\right)^{t+1}\left(\mathcal{L}(\boldsymbol{\theta}_0)-\mathcal{L}_\star\right)\\&\leq\frac{\kappa\zeta_\star^2}{2L}\frac{1}{1-\left(1-\frac{\mu}{L}\left(1-\kappa(P-1)\right)\right)}+\left(1-\frac{\mu}{L}\left(1-\kappa(P-1)\right)\right)^{t+1}\left(\mathcal{L}(\boldsymbol{\theta}_0)-\mathcal{L}_\star\right)\\&=\frac{\kappa\zeta_\star^2}{2\mu\left(1-\kappa(P-1)\right)}+\left(1-\frac{\mu}{L}\left(1-\kappa(P-1)\right)\right)^{t+1}\left(\mathcal{L}(\boldsymbol{\theta}_0)-\mathcal{L}_\star\right).\end{aligned}$$

Using the fact that $(1+x)^n\leq e^{nx}$ for all $x\in\mathbb{R}$ and substituting $t=T_{\mathrm{I}}-1$ yields

$$\mathcal{L}(\boldsymbol{\theta}_{T_{\mathrm{I}}})-\mathcal{L}_\star\leq\frac{\kappa\zeta_\star^2}{2\mu\left(1-\kappa(P-1)\right)}+e^{-\frac{\mu}{L}\left(1-\kappa(P-1)\right)T_{\mathrm{I}}}\left(\mathcal{L}(\boldsymbol{\theta}_0)-\mathcal{L}_\star\right).$$

Now, for simplicity, denote $\boldsymbol{\theta}_{\mathrm{TGD}}\coloneqq\boldsymbol{\theta}_{T_{\mathrm{I}}}$. Then, remarking that $\frac{\kappa}{1-\kappa(P-1)}\leq 45\frac{f}{n}$ and $1-\kappa(P-1)\geq\frac{1}{2}$ when $\frac{f}{n}\leq\min\left\{\frac{1}{3},\frac{12}{5(P-1)}\right\}$ yields that

$$\mathcal{L}(\boldsymbol{\theta}_{\mathrm{TGD}})-\mathcal{L}_\star\leq\frac{45}{2\mu}\frac{f}{n}\zeta_\star^2+e^{-\frac{\mu}{2L}T_{\mathrm{I}}}\left(\mathcal{L}(\boldsymbol{\theta}_0)-\mathcal{L}_\star\right).\tag{17}$$

Recall that $\mathcal{L}$ is $L$-smooth by assumption. We now analyze $T$ iterations of fine-tuning with gradient descent, i.e., the sequence of iterations given by $\boldsymbol{\theta}'_{t+1}=\boldsymbol{\theta}'_t-\gamma\nabla\mathcal{L}(\boldsymbol{\theta}'_t)$, for $t\in\{0,\ldots,T-1\}$, with $\boldsymbol{\theta}'_0=\boldsymbol{\theta}_{\mathrm{TGD}}$. Hence, by the smoothness assumption, we have

$$\mathcal{L}(\boldsymbol{\theta}'_{t+1})-\mathcal{L}(\boldsymbol{\theta}'_t)\leq-\gamma\left\|\nabla\mathcal{L}(\boldsymbol{\theta}'_t)\right\|^2+\frac{1}{2}\gamma^2L\left\|\nabla\mathcal{L}(\boldsymbol{\theta}'_t)\right\|^2.\tag{18}$$

Substituting $\gamma=\frac{1}{L}$ and using the PL inequality, we get

$$\mathcal{L}(\boldsymbol{\theta}'_{t+1})-\mathcal{L}(\boldsymbol{\theta}'_t)\leq-\frac{1}{2L}\left\|\nabla\mathcal{L}(\boldsymbol{\theta}'_t)\right\|^2\leq-\frac{\mu}{L}(\mathcal{L}(\boldsymbol{\theta}'_t)-\mathcal{L}_\star).\tag{19}$$

Rearranging terms then yields

$$\mathcal{L}(\boldsymbol{\theta}'_{t+1})-\mathcal{L}_\star\leq(1-\frac{\mu}{L})(\mathcal{L}(\boldsymbol{\theta}'_t)-\mathcal{L}_\star).$$

By recursively applying the above for times indices in $\{0,\ldots,T-1\}$, and using the fact that $(1+x)^n\leq e^{nx}$ for all $x\in\mathbb{R}$, we obtain

$$\mathcal{L}(\boldsymbol{\theta}'_T)-\mathcal{L}_\star\leq e^{-\frac{\mu}{L}T}(\mathcal{L}(\boldsymbol{\theta}'_0)-\mathcal{L}_\star)=e^{-\frac{\mu}{L}T}(\mathcal{L}(\boldsymbol{\theta}_{\mathrm{TGD}})-\mathcal{L}_\star).$$

Now, plugging (17) in the above, we obtain

$$\mathcal{L}(\boldsymbol{\theta}'_T)-\mathcal{L}_\star\leq e^{-\frac{\mu}{L}T}\left[\frac{45}{2\mu}\frac{f}{n}\zeta_\star^2+e^{-\frac{\mu}{2L}T_{\mathrm{I}}}\left(\mathcal{L}(\boldsymbol{\theta}_0)-\mathcal{L}_\star\right)\right]=\frac{45e^{-\frac{\mu}{L}T}}{2\mu}\frac{f}{n}\zeta_\star^2+e^{-\frac{\mu}{L}(T+\frac{T_{\mathrm{I}}}{2})}\left(\mathcal{L}(\boldsymbol{\theta}_0)-\mathcal{L}_\star\right).$$

Therefore, if $T=\mathcal{O}(\frac{L}{\mu}\log(\frac{f}{n}\frac{\zeta_\star^2}{\mu}\frac{1}{\varepsilon}))$ and $T_{\mathrm{I}}=\mathcal{O}(\frac{L}{\mu}\log(\frac{\mathcal{L}(\boldsymbol{\theta}_0)-\mathcal{L}_\star}{\varepsilon}))$, then the procedure satisfies $\varepsilon$-approximate unlearning. □

## B. Training details

*Table 1.* Training Parameters & Steps

| | Batch Size | LR | Optimizer | Learning Steps | Unlearning steps | Dropout |
|---|---|---|---|---|---|---|
| **MNIST** | 250 | 3e-4 | Adam | 100,000 | 100,000 | 0.25 |
| **CIFAR-10** | 250 | 3e-4 | Adam | 100,000 | 100,000 | 0.25 |

Experiments were performed on two types of architectures, a CNN and a ResNet-18 architecture. The CNN had 4 convolutional layers with max pooling, followed by 3 fully-connected layers. The ResNet-18 architecture is the same as in (Sai Abhishek, 2022). Training parameters can be found in Table 1. The same unlearning parameters were used for all three types of unlearning that we tested experimentally.

## C. Additional Experiments

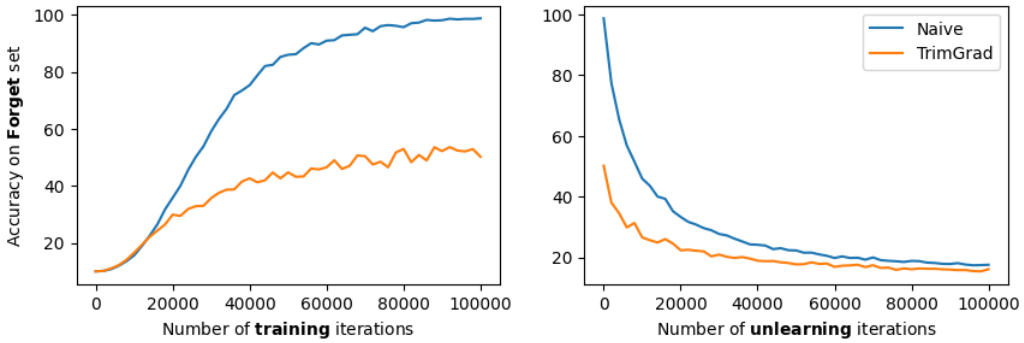


*Figure 2.* Accuracy on the forget set during training (left) and unlearning (right) naive training (Naive), robust training (TRIMGRAD), and unlearning by retraining from scratch (Scratch), on CIFAR-10 with a convolutional neural network.

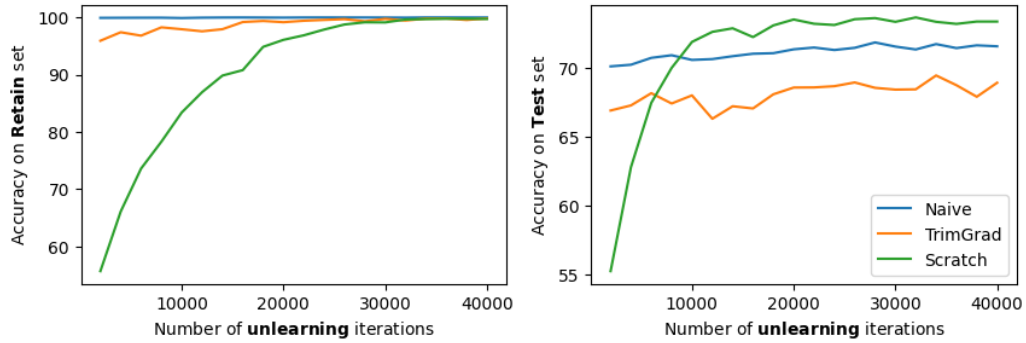


*Figure 3.* Accuracy on the retain set (left) and accuracy on the test set (right) during unlearning after naive training (Naive), robust training (TRIMGRAD), and unlearning by retraining from scratch (Scratch), on CIFAR-10 with a convolutional neural network.

## D. Sequential Removal Requests

We can adapt TRIMGRAD to sequential removal requests by sequentially removing the forget set and repeating the TRIMGRAD procedure on the retain set, as summarized in Algorithm 2. Each model obtained prior to removal can be used in parallel for unlearning, e.g., via fine-tuning.

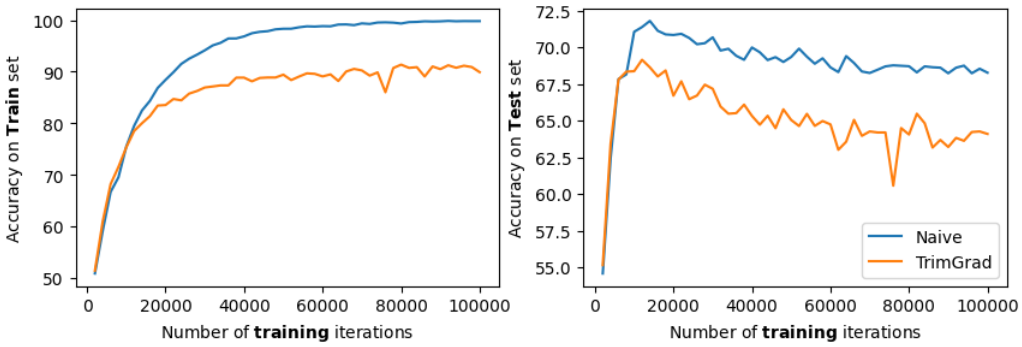


*Figure 4.* Accuracy on the train set (left) and accuracy on the test set (right) during training with naive training (Naive), robust training (TRIMGRAD), on CIFAR-10 with a convolutional neural network.

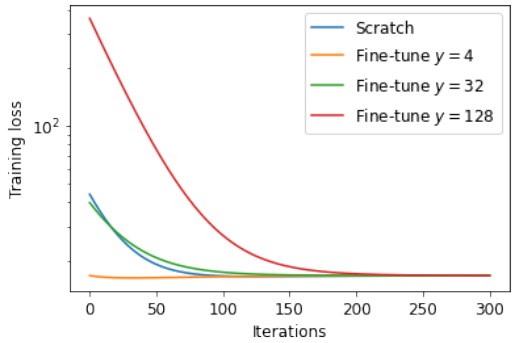


*Figure 5.* Training loss vs. number of iterations during unlearning, after naive training and naive fine-tuning (Fine-tune), compared to training from scratch (Scratch) with different values of the label of the (outlier) forget data point.

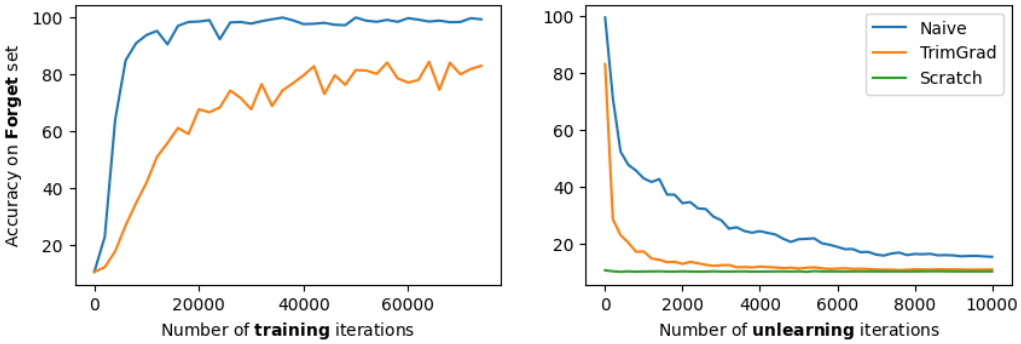


*Figure 6.* Accuracy on the forget set during training (left) and unlearning (right) naive training (Naive), robust training (TRIMGRAD), and unlearning by retraining from scratch (Scratch), on MNIST with a fully-connected two-layer neural network.

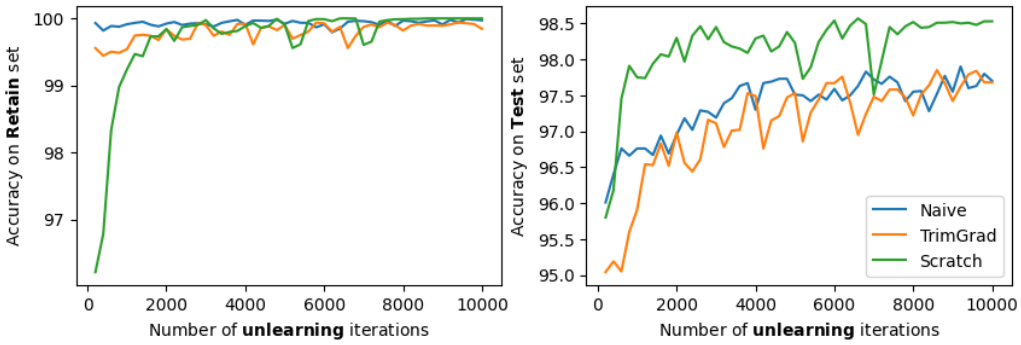


*Figure 7.* Accuracy on the retain set (left) and accuracy on the test set (right) during unlearning after naive training (Naive), robust training (TRIMGRAD), and unlearning by retraining from scratch (Scratch), on MNIST with a fully-connected two-layer neural network.

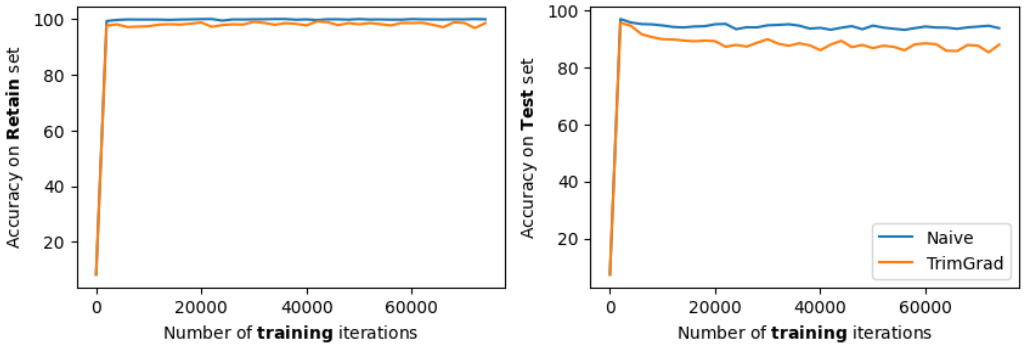


*Figure 8.* Accuracy on the retain set (left) and accuracy on the test set (right) during training with naive training (Naive), robust training (TRIMGRAD), on MNIST with a fully-connected two-layer neural network.

**Algorithm 2** TRIMGRAD WITH SEQUENTIAL REMOVAL REQUESTS

**Input:** Initial model $\boldsymbol{\theta}_0$, sequence of forget sets $\mathcal{S}_f^0, \ldots, \mathcal{S}_f^{K-1}$ of sizes $f_1, \ldots, f_K$, learning rate $\gamma$, and number of steps $T$.
**for** $k = 0 \ldots K-1$ **do**
  **if** $k = 0$ **then**
    $\boldsymbol{\theta}_0^k = \boldsymbol{\theta}_0$
    $\mathcal{S}^k = \mathcal{S}$
  **end**
  **else**
    $\boldsymbol{\theta}_0^k = \boldsymbol{\theta}_T^{k-1}$
    $\mathcal{S}^k = \mathcal{S}^{k-1} \setminus \mathcal{S}_f^{k-1}$
  **end**
  Update training set: $\mathcal{S}^k = \mathcal{S}^{k-1}$
  **for** $t = 0 \ldots T-1$ **do**
    Compute the trimmed average gradient: $\mathbf{r}_t^k = \mathrm{TM}_{f_k}\left(\left\{\nabla \ell(\boldsymbol{\theta}_t^k; \mathbf{z}) \colon \mathbf{z} \in \mathcal{S}^k\right\}\right)$
    Update the model: $\boldsymbol{\theta}_{t+1}^k = \boldsymbol{\theta}_t^k - \gamma \mathbf{r}_t^k$
  **end**
**end**