Unlearning Tabular Data Without a “Forget Set”

Aviraj Newatia[1,2] **Michael Cooper**[1,2] **Rahul G. Krishnan**[1,2]
[1]University of Toronto [2]Vector Institute
`{avinewatia,coopermj,rahulgk}@cs.toronto.edu`

Abstract

Machine unlearning is the process of removing the influence of some subset of the training data from the parameters of a previously-trained model. Existing methods typically require direct access to the “forget set" – the subset of training data to be forgotten by the model. This limitation impedes privacy, as organizations need to retain user data for the sake of unlearning when a request for deletion is made, rather than being able to delete it immediately. We introduce RELOAD, an approximate unlearning algorithm that leverages ideas from gradient-based unlearning and neural network sparsity to achieve blind unlearning in settings of tabular data. The method serially applies an ascent step with targeted parameter re-initialization and fine-tuning, and on empirical unlearning tasks, RELOAD often approximates the behaviour of a from-scratch retrained model better than approaches that leverage the forget set. Empirical results highlight how RELOAD has the potential to improve privacy-preserving machine learning in the tabular setting.

1 Introduction

Machine unlearning poses the problem of removing the influence of certain instances in the training data on a given statistical model [4]. Motivated by “right to be forgotten" provisions [7], methods in machine unlearning aim to provide efficient means to “forget" specific data points from a trained model without requiring that it be retrained from scratch. Machine unlearning in the tabular data regime remains under-explored – to our knowledge having been studied only in Warnecke et al. [23] – despite being a common data modality on which effective unlearning may be desirable [13, 27]. As larger models for tabular data become more prevalent [26, 14, 11], the need to unlearn specific data instances without retraining from scratch is increasingly important.

Contemporary unlearning methods generally require explicit access to the so-called “forget set” – the subset of training data to be forgotten by the model. For example, one approach entails performing steps of gradient ascent on the loss landscape characterized by the forget set in order to remove its influence on the model weights [22]. However, the reliance of these methods on the forget set introduces a tension in the context of preserving user privacy: in order to enable unlearning, organizations must retain the complete original set of user data on which the model was trained. Retaining this data, even for the purpose of unlearning, can expose organizations and individuals to risks associated with data breaches or unauthorized access. To bridge this gap, there is a clear need for unlearning methods that operate without requiring access to the forget set.

This work presents an algorithm for tabular machine unlearning in the absence of an explicitly defined forget set; a setting we establish as “blind unlearning.” Our method, RELOAD, assumes that the modeller only has access to (a) a model trained on a dataset $\mathcal{D}$, (b) the “retain set," $\mathcal{D}_{retain} \triangleq \mathcal{D} \backslash \mathcal{D}_{forget}$, and (c) cached gradients from the last iteration of training on $\mathcal{D}$. Notable in its absence from these requirements is the forget set – this means that RELOAD allows the modeller to delete instances in $\mathcal{D}_{forget}$ at the conclusion of model training without compromising his or her ability to perform machine unlearning downstream.

Our work makes the following contributions. (1) We first introduce RELOAD, an algorithm for machine unlearning that does not require access to the “forget set." (2) Using an established model of tabular data [1], we then demonstrate that RELOAD permits effective unlearning of data instances in the tabular regime. (3) We extend our method to consider *feature unlearning* – e.g., sensitive attributes that pertain to all samples in the data. We show how RELOAD permits feature unlearning, and employ TabNet's saliency weights to showcase how feature unlearning with RELOAD changes the features to which the model attends at each decision step.

2 Background and Related Work

Let $\mathcal{D} = \{(X_i, Y_i)\}_{i=1,\ldots,N}$ represent a collection of i.i.d. data, where $X \in \mathcal{X}$ represents input covariates and $Y \in \mathcal{Y}$ represents labels for supervised learning. Then, for some class of models $\mathcal{M}$, let θ^* represent the parameters that minimize the empirical loss with respect to training data $\mathcal{D}$. We denote an instantiation of $\mathcal{M}$ trained on $\mathcal{D}$ as $\mathcal{M}^{(\theta^*)}$. After $\mathcal{M}^{(\theta^*)}$ is trained, assume that some information is removed from $\mathcal{D}$ to yield $\mathcal{D}_{retain}$ (e.g., deleting instances or features from $\mathcal{D}$). Then, $\theta^{\sim}$ represents the parameters that minimize the empirical loss with respect to $\mathcal{D}_{retain}$.

The goal of approximate unlearning methods like RELOAD is to efficiently learn an approximation of $\mathcal{M}^{(\theta^{\sim})}$. The classical setting assumes that the modeller has access to the trained model $\mathcal{M}^{(\theta^*)}$, the training dataset $\mathcal{D}$, the remaining data $\mathcal{D}_{retain}$, and the forget set $\mathcal{D}_{forget}$ [5], whereas RELOAD aims to remove the dependence on $\mathcal{D}_{forget}$.

Machine Unlearning. Many existing approximate unlearning algorithms perform an optimization procedure on $\mathcal{M}^{(\theta^*)}$ using the forget set $\mathcal{D}_{forget}$ and the retain set $\mathcal{D}_{retain}$. One simple standard approach applies gradient ascent on the loss with respect to $\mathcal{D}_{forget}$, in order to undo the parameter updates induced by those instances during training [12, 22]. Another gradient-based approach leverages a teacher-student method: SCalable Remembering and Unlearning unBound (SCRUB) distills a student model from a teacher trained on $\mathcal{D}$, but the student learns to selectively disobey the teacher by directly maximizing the loss on $\mathcal{D}_{forget}$ [16].

Weight Saliency-Based Approximate Unlearning. Another class of approximate unlearning methods derives from the hypothesis that identifiable substructures in neural networks often correspond to different subsets of the training data [18]. These methods leverage ideas from neural sparsity [10, 6] to perform unlearning on specific parameters. Saliency unlearning (SalUn) uses a threshold on $\nabla_\theta \mathcal{L}(\mathcal{D}_{forget})$ to identify parameters containing the most signal about $\mathcal{D}_{forget}$ and focuses model updates on these parameters [8]. Selective Synaptic Dampening (SSD) [9] avoids gradient steps by scaling parameters based on their Fisher Information Matrix importance scores.

3 The RELOAD Algorithm

RELOAD implements unlearning by using the gradients of the entire training set to identify weights that most strongly correspond to items in the forget set. By selectively retraining these parameters, RELOAD permits machine unlearning without requiring explicit access to the forget set. We argue that preserving the summed gradients of the entire training set – which includes those of the forget set – is a modest operation that can be performed at the conclusion of model training. Moreover, we propose that these gradients do not permit recoverability of the data in question (see Appendix A.3). Intuition for the algorithm is provided in Appendix A.2.

Direction of Movement. The challenge of blind unlearning is that taking repeated gradients of $\mathcal{L}(\mathcal{D}_{forget})$ is impossible without access to $\mathcal{D}_{forget}$. However, from cached gradients of $\mathcal{D}$ at the conclusion of model training, $\nabla_\theta \mathcal{L}(\mathcal{D})$, we infer $\nabla_\theta \mathcal{L}(\mathcal{D}_{forget})$ by the linearity of differentiation, $\nabla_\theta \mathcal{L}(\mathcal{D}_{forget}) = \nabla_\theta \mathcal{L}(\mathcal{D}) - \nabla_\theta \mathcal{L}(\mathcal{D}_{retain})$. Therefore, a gradient-based descent update in the direction of $\nabla_\theta \mathcal{L}(\mathcal{D}_{forget})$ moves the model parameters such that they better fit to $\mathcal{D}_{forget}$; because our goal is *unlearning* $\mathcal{D}_{forget}$, RELOAD instead begins with a single *ascent* update in this direction.

Targeted Parameter Adjustments. As a single ascent step is not sufficient, we subsequently re-initialize selected network parameters that contain a disproportionate amount of the necessary information to characterize instances in $\mathcal{D}_{forget}$. Consider the gradient $\nabla_{\theta_k} \mathcal{L}(\mathcal{D}_{forget})$, the gradient of the loss with respect to instances in $\mathcal{D}_{forget}$ and with respect to a particular parameter θ_k. If this gradient is small, it means that θ_k is well-optimized for instances in $\mathcal{D}_{forget}$. The relative magnitude of $\nabla_{\theta_k} \mathcal{L}(\mathcal{D}_{forget})$ to $\nabla_{\theta_k} \mathcal{L}(\mathcal{D})$ is a meaningful representation of the extent to which θ_k is responsible for characterizing information about $\mathcal{D}_{forget}$. We call this the *knowledge value* of parameter θ_k, and write $KV_{\theta_k} \triangleq \frac{|\nabla_{\theta_k} \mathcal{L}(\mathcal{D}_{forget})| + \epsilon}{|\nabla_{\theta_k} \mathcal{L}(\mathcal{D})| + \epsilon} = \frac{|\nabla_{\theta_k} \mathcal{L}(\mathcal{D}) - \nabla_{\theta_k} \mathcal{L}(\mathcal{D}_{retain})| + \epsilon}{|\nabla_{\theta_k} \mathcal{L}(\mathcal{D})| + \epsilon}$, where ϵ is

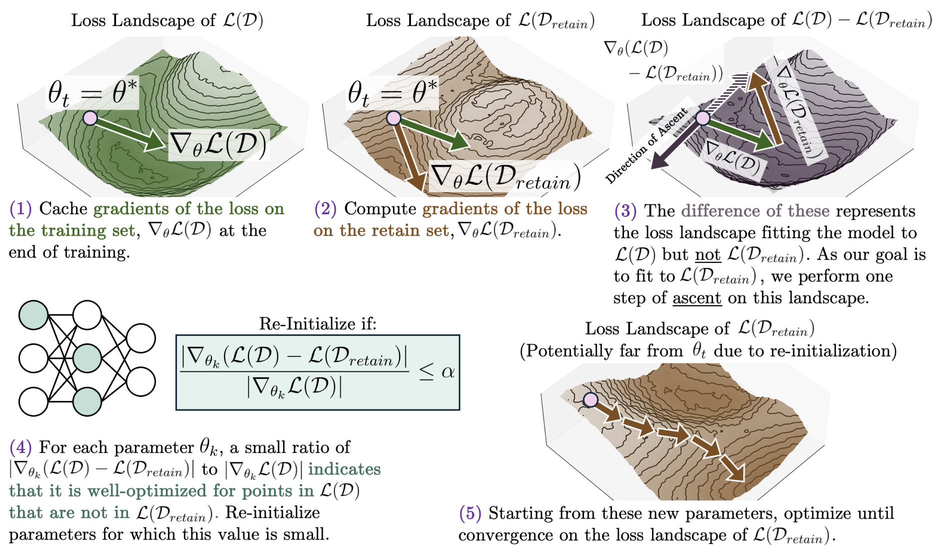
Loss Landscape of $\mathcal{L}(\mathcal{D})$
$\theta_t = \theta^*$
$\nabla_\theta \mathcal{L}(\mathcal{D})$
(1) Cache gradients of the loss on the training set, $\nabla_\theta \mathcal{L}(\mathcal{D})$ at the end of training.
Loss Landscape of $\mathcal{L}(\mathcal{D}_{retain})$
$\theta_t = \theta^*$
$\nabla_\theta \mathcal{L}(\mathcal{D}_{retain})$
(2) Compute gradients of the loss on the retain set, $\nabla_\theta \mathcal{L}(\mathcal{D}_{retain})$.
Loss Landscape of $\mathcal{L}(\mathcal{D}) - \mathcal{L}(\mathcal{D}_{retain})$
$\nabla_\theta(\mathcal{L}(\mathcal{D}) - \mathcal{L}(\mathcal{D}_{retain}))$
$\nabla_\theta \mathcal{L}(\mathcal{D}_{retain})$
$\nabla_\theta \mathcal{L}(\mathcal{D})$
Direction of Ascent
(3) The difference of these represents the loss landscape fitting the model to $\mathcal{L}(\mathcal{D})$ but not $\mathcal{L}(\mathcal{D}_{retain})$. As our goal is to fit to $\mathcal{L}(\mathcal{D}_{retain})$, we perform one step of ascent on this landscape.
Re-Initialize if:
$\frac{|\nabla_{\theta_k}(\mathcal{L}(\mathcal{D}) - \mathcal{L}(\mathcal{D}_{retain})|}{|\nabla_{\theta_k}\mathcal{L}(\mathcal{D})|} \leq \alpha$
(4) For each parameter θ_k, a small ratio of $|\nabla_{\theta_k}(\mathcal{L}(\mathcal{D}) - \mathcal{L}(\mathcal{D}_{retain})|$ to $|\nabla_{\theta_k}\mathcal{L}(\mathcal{D})|$ indicates that it is well-optimized for points in $\mathcal{L}(\mathcal{D})$ that are not in $\mathcal{L}(\mathcal{D}_{retain})$. Re-initialize parameters for which this value is small.
Loss Landscape of $\mathcal{L}(\mathcal{D}_{retain})$
(Potentially far from θ_t due to re-initialization)
(5) Starting from these new parameters, optimize until convergence on the loss landscape of $\mathcal{L}(\mathcal{D}_{retain})$.

Figure 1: The RELOAD algorithm marries a gradient-based unlearning step modified for the blind unlearning setting (Steps **(1)** through **(3)**) with a weight saliency-based selective re-initialization (Step **(4)**) and subsequent fine-tuning (Step **(5)**). Because the blind unlearning setting prohibits taking gradients with respect to $\mathcal{D}_{forget}$, RELOAD exploits the linearity of differentiation to treat $\nabla_\theta(\mathcal{L}(\mathcal{D}) - \mathcal{L}(\mathcal{D}_{retain}))$ as a proxy for $\nabla_\theta \mathcal{L}(\mathcal{D}_{forget})$ at the location in parameter space corresponding to θ_t. This allows us to apply one ascent step in this direction. Intuitively, this update in Step **(3)** removes information about $\mathcal{D}_{forget}$ from *all* network parameters, while the re-initialization in Step **(4)** re-initialises those parameters with a uniquely strong correspondence to $\mathcal{D}_{forget}$ (for which a single ascent step will not fully remove this information).

a small Laplace smoothing constant. A small knowledge value characterizes a parameter that is knowledgeable about $\mathcal{D}_{forget}$; it is these parameters that we reinitialize.

4 Experimental Configuration

We consider a selection of tasks to demonstrate RELOAD. We first randomly assign 30% of samples in the training data to $\mathcal{D}_{forget}$ to highlight *item unlearning*. Next, we study the unlearning of a feature from the training data. We compare RELOAD against several baselines from the machine unlearning literature (discussed in greater detail in Appendix A.1). All our our experiments (both with RELOAD and with the baselines) use a TabNet model configured and trained according to reference implementation. All models were trained on the standard Adult Census Income dataset used to benchmark the original TabNet model [15, 3]. All models were trained (or retrained) for 60 epochs on the training set. Evaluation metrics are shown in Table 1.

Algorithm 1 The RELOAD Algorithm for Blind Unlearning

```
1: Input: $\mathcal{M}^{(\theta^*)}$, cached $\nabla_\theta \mathcal{L}(\mathcal{D})$, $\mathcal{D}_{retain}$
2: Parameters: $\eta_p$: priming step learning rate, $\epsilon$: noise parameter, $\alpha$: reset proportion
3: Output: Trained model approximating $\mathcal{M}^{(\theta^\sim)}$
4:
5: procedure RELOAD($\mathcal{M}^{(\theta^*)}$, $\nabla_\theta \mathcal{L}(\mathcal{D}; \mathcal{M}^{(\theta^*)})$, $\mathcal{D}_{retain}$)
6:     $\theta' \leftarrow \theta^* + \eta_p \nabla_\theta(\mathcal{L}(\mathcal{D}) - \mathcal{L}(\mathcal{D}_{retain}))$        ▷ Step (2 – 3) (Fig. 1)
7:     KV $\leftarrow \left\{ \frac{|\nabla_{\theta_k}\mathcal{L}(\mathcal{D}) - \nabla_{\theta_k}\mathcal{L}(\mathcal{D}_{retain})| + \epsilon}{|\nabla_{\theta_k}\mathcal{L}(\mathcal{D})| + \epsilon} \right\}_{\theta_k \in \theta}$        ▷ Step (3) (Fig. 1)
8:     for $\theta_k \in \theta'$ do
9:         $\theta'_k \leftarrow$ INITIALIZE($\cdot$) if QUANTILE$_{KV}(KV_{\theta_k}) \leq \alpha$        ▷ Step (4) (Fig. 1)
10:    end for
11:    Train $\mathcal{M}^{(\theta')}$ to convergence on $\mathcal{D}_{retain}$        ▷ Step (5) (Fig. 1)
12: end procedure
```

5 Results

Table 2 presents an empirical evaluation of RELOAD in the tabular setting. The *item unlearning* results (*Left*) show that RELOAD yields a model that is close to the gold-standard retrained model in ΔFA, ΔFE, and ΔFMIA. Furthermore, our method achieves high RA, indicating that the model keeps utility on the retain set. In the *feature unlearning* setting, (*Right*) we observe that the original model exhibits a significant drop in accuracy (from 0.84 to 0.73); however, applying RELOAD largely corrects for this, returning training-set and and test-set accuracies comparable to those of a model

Statistic	**Abbr.**	**Description**
Retain-Set Accuracy (↑)	RA	Model accuracy on the retain-set. A higher accuracy indicates that the unlearning process did not negatively impact the model's performance on the retained data.
Diff. in Forget-Set Accuracy (↓)	ΔFA	The change in accuracy on the forget set between the current model and a baseline model retrained from scratch without the forget set. A smaller difference, approaching the accuracy of the retrained model, indicates that the unlearning method has been more effective in "forgetting" the forget set.
Diff. in Forget-Set Error (↓)	ΔFE	The reduction in error on the forget set between the current model and a baseline model retrained from scratch without the forget set. A smaller difference, approaching the error of the retrained model, signifies that the unlearning method has been more effective at "forgetting" the forget set.
Diff. in Forget-Set MIA Success Rate (↓)	ΔFMIA	Difference in success rate of a membership inference attack (MIA) on the forget set between the current model and a baseline model retrained from scratch without the forget set. In this work, we use the attack and implementation from Kurmanji et al. [16]. A success rate close to the retrained model's implies the forgotten data is indistinguishable to an MIA on in-distribution data that the model was not trained on.
Cost (↓)	Cost	Ratio of the runtime of the unlearning method to the runtime of retraining a baseline model from scratch without the forget set. A lower cost indicates a more computationally efficient method.
Original Acc. (↑)	OA	Accuracy of the model on the original training set $\mathcal{D}$ containing all samples including the ones that are to be forgotten.
Test Acc. (↑)	TA	Accuracy of the model on a test dataset.

Table 1: Summary of evaluation statistics that we use in comparing RELOAD to baseline algorithms.

naively retrained from scratch. Interestingly, we observe that Fine-Tuning achieves higher accuracy than RELOAD on the retain-set. More broadly, in contrast with preliminary experiments we have conducted on image data (omitted for brevity), we observe that naive fine tuning appears to yield far better performance in both *item unlearning* and *feature unlearning* in the tabular setting. We find this intriguing phenomenon worthy of further study – perhaps there are unique elements within the structure of tabular data that explains this dynamic.

We next visualise the feature dependence from the models yielded by all of these algorithms in Figure 2, which shows that, when viewed from the perspective of the feature importance weights in TabNet, RELOAD cleanly removes the influence of the target feature on model prediction.

	Forgetting a Random 30% of the Data					**Forgetting a Random Feature**			
Method	RA (↑)	ΔFA (↓)	ΔFE (↓)	ΔFMIA (↓)	Cost (↓)	OA (↑)	RA (↑)	TA (↑)	Cost (↓)
Original	N/A	N/A	N/A	N/A	N/A	$0.84_{\pm 0.00}$	$0.73_{\pm 0.07}$	$0.82_{\pm 0.01}$	N/A
Retrain	$0.84_{\pm 0.01}$	$0.82_{\pm 0.01}$	$0.48_{\pm 0.00}$	$0.51_{\pm 0.13}$	$1.00_{\pm 0.00}$	$0.77_{\pm 0.09}$	$0.84_{\pm 0.01}$	$0.75_{\pm 0.09}$	$1.00_{\pm 0.00}$
RELOAD	$\mathbf{0.84_{\pm 0.01}}$	$0.02_{\pm 0.01}$	$\mathbf{0.00_{\pm 0.00}}$	$\mathbf{0.00_{\pm 0.00}}$	$0.29_{\pm 0.18}$	$\mathbf{0.78_{\pm 0.03}}$	$0.81_{\pm 0.01}$	$\mathbf{0.77_{\pm 0.03}}$	$0.40_{\pm 0.27}$
GA	$0.38_{\pm 0.22}$	$0.54_{\pm 0.04}$	$0.55_{\pm 0.04}$	$0.06_{\pm 0.07}$	$\mathbf{0.13_{\pm 0.02}}$	$0.35_{\pm 0.16}$	$0.48_{\pm 0.17}$	$0.35_{\pm 0.16}$	$\mathbf{0.30_{\pm 0.12}}$
FT	$0.83_{\pm 0.01}$	$\mathbf{0.01_{\pm 0.01}}$	$0.01_{\pm 0.00}$	$0.05_{\pm 0.06}$	$0.53_{\pm 0.07}$	$0.77_{\pm 0.09}$	$\mathbf{0.84_{\pm 0.00}}$	$0.76_{\pm 0.09}$	$0.79_{\pm 0.20}$
GA FT	$0.77_{\pm 0.03}$	$0.06_{\pm 0.02}$	$0.05_{\pm 0.02}$	$0.13_{\pm 0.08}$	$0.71_{\pm 0.09}$	$0.77_{\pm 0.03}$	$0.79_{\pm 0.02}$	$0.76_{\pm 0.03}$	$1.09_{\pm 0.31}$

Table 2: Results highlighting the performance of RELOAD on the tabular Census Income dataset. *(Left)* Observe how, when used to forget a random 30% of the data, RELOAD achieves comparable performance to a model retrained on $\mathcal{D}_{retain}$, and outperforms several baseline methods that rely on $\mathcal{D}_{forget}$. *(Right)* When used to forget a randomly selected feature from the dataset, RELOAD largely corrects for the drop in accuracy experienced by the original model. Each of the numbers in this table corresponds to the mean value over five randomly-reinitialized experiments (wherein the 30% of data, and feature to be forgotten, may vary).

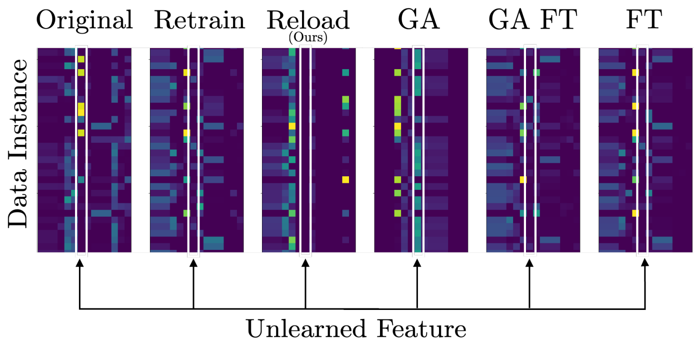

Figure 2: Feature importance masks obtained from the first decision step of TabNet, under various unlearning schemes. Observe how naive Gradient Ascent (undesirably) retains the influence of this feature within the model, while Retraining, RELOAD, Gradient Ascent (with Fine Tuning), and Fine Tuning do not. This highlights how from the perspective of feature importance, RELOAD enables effective, efficient feature unlearning.

6 Conclusion

We introduce RELOAD, a novel algorithm for machine unlearning that effectively removes data influence in tabular models without requiring access to a "forget set." Our experiments with the TabNet model illustrate its efficiency and effectiveness, paving the way for further exploration of unlearning in the oft-unexplored tabular setting. Future research can build on these findings by exploring additional architectures and datasets, or by designing novel unlearning methods to more directly exploit the unique structure of tabular data in the unlearning process.

References

[1] S. Ö. Arik and T. Pfister. Tabnet: Attentive interpretable tabular learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 6679–6687, 2021.

[2] D. Arpit, S. Jastrzebski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. C. Courville, Y. Bengio, and S. Lacoste-Julien. A closer look at memorization in deep networks. In *International Conference on Machine Learning*, 2017. URL `https://api.semanticscholar.org/CorpusID:11455421`.

[3] A. Asuncion, D. Newman, et al. Uci machine learning repository, 2007.

[4] L. Bourtoule, V. Chandrasekaran, C. A. Choquette-Choo, H. Jia, A. Travers, B. Zhang, D. Lie, and N. Papernot. Machine unlearning. 12 2019. URL `http://arxiv.org/abs/1912.03817`.

[5] Y. Cao and J. Yang. Towards making systems forget with machine unlearning. In *2015 IEEE Symposium on Security and Privacy*, pages 463–480, 2015. doi: 10.1109/SP.2015.35.

[6] A. Chen, R. I. Shi, X. Gao, R. Baptista, and R. G. Krishnan. Structured neural networks for density estimation and causal inference. *Advances in Neural Information Processing Systems*, 36, 2024.

[7] European Parliament and Council of the European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council. URL `https://data.europa.eu/eli/reg/2016/679/oj`.

[8] C. Fan, J. Liu, Y. Zhang, E. Wong, D. Wei, and S. Liu. Salun: Empowering machine unlearning via gradient-based weight saliency in both image classification and generation, 2023. URL `https://arxiv.org/abs/2310.12508`.

[9] J. Foster, S. Schoepf, and A. Brintrup. Fast machine unlearning without retraining through selective synaptic dampening. *ArXiv*, abs/2308.07707, 2023. URL `https://api.semanticscholar.org/CorpusID:260900355`.

[10] J. Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

[11] J. Gardner, J. C. Perdomo, and L. Schmidt. Large scale transfer learning for tabular data via language modeling. *arXiv preprint arXiv:2406.12031*, 2024.

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

[13] M. Hernadez, G. Epelde, A. Alberdi, R. Cilla, and D. Rankin. Synthetic tabular data evaluation in the health domain covering resemblance, utility, and privacy dimensions. *Methods of information in medicine*, 62(S 01):e19–e38, 2023.

[14] M. Kayali, A. Lykov, I. Fountalis, N. Vasiloglou, D. Olteanu, and D. Suciu. Chorus: foundation models for unified data discovery and exploration. *arXiv preprint arXiv:2306.09610*, 2023.

[15] R. Kohavi. Census Income. UCI Machine Learning Repository, 1996. DOI: https://doi.org/10.24432/C5GP7S.

[16] M. Kurmanji, P. Triantafillou, J. Hayes, and E. Triantafillou. Towards unbounded machine unlearning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL `https://openreview.net/forum?id=OveBaTtUAT`.

[17] M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation*, 24:109–165, 1989. URL `https://api.semanticscholar.org/CorpusID:61019113`.

[18] J. Pfeiffer, S. Ruder, I. Vulić, and E. Ponti. Modular deep learning. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL `https://openreview.net/forum?id=z9EkXfvxta`. Survey Certification.

[19] M. R. I. Rabin, A. Hussain, M. A. Alipour, and V. J. Hellendoorn. Memorization and generalization in neural code intelligence models. *Information and Software Technology*, 153:107066, Jan. 2023. ISSN 0950-5849. doi: 10.1016/j.infsof.2022.107066. URL `http://dx.doi.org/10.1016/j.infsof.2022.107066`.

[20] D. Smilkov, N. Thorat, B. Kim, F. B. Viégas, and M. Wattenberg. Smoothgrad: removing noise by adding noise. *ArXiv*, abs/1706.03825, 2017. URL `https://api.semanticscholar.org/CorpusID:11695878`.

[21] D. Smilkov, N. Thorat, B. Kim, F. B. Viégas, and M. Wattenberg. Smoothgrad: removing noise by adding noise. *ArXiv*, abs/1706.03825, 2017. URL `https://api.semanticscholar.org/CorpusID:11695878`.

[22] A. Thudi, G. Deza, V. Chandrasekaran, and N. Papernot. Unrolling sgd: Understanding factors influencing machine unlearning. In *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&amp;P)*, pages 303–319, Los Alamitos, CA, USA, jun 2022. IEEE Computer Society. doi: 10.1109/EuroSP53844.2022.00027. URL `https://doi.ieeecomputersociety.org/10.1109/EuroSP53844.2022.00027`.

[23] A. Warnecke, L. Pirch, C. Wressnegger, and K. Rieck. Machine unlearning of features and labels. *ArXiv*, abs/2108.11577, 2021. URL `https://api.semanticscholar.org/CorpusID:237303799`.

[24] A. Warnecke, L. Pirch, C. Wressnegger), and K. Rieck. Machine unlearning of features and labels. In *Proceedings 2023 Network and Distributed System Security Symposium*, NDSS 2023. Internet Society, 2023. doi: 10.14722/ndss.2023.23087. URL `http://dx.doi.org/10.14722/ndss.2023.23087`.

[25] H. Xu, X. Liu, W. Wang, Z. Liu, A. K. Jain, and J. Tang. How does the memorization of neural networks impact adversarial robust models? In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, page 2801–2812, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701030. doi: 10.1145/3580305.3599381. URL `https://doi.org/10.1145/3580305.3599381`.

[26] S. Yak, Y. Dong, J. Gonzalvo, and S. Arik. Ingestables: Scalable and efficient training of llm-enabled tabular foundation models. In *NeurIPS 2023 Second Table Representation Learning Workshop*, 2023.

[27] E. Zavitsanos, D. Mavroeidis, E. Spyropoulou, M. Fergadiotis, and G. Paliouras. Entrant: A large financial dataset for table understanding. *Scientific Data*, 11(1):876, 2024.

A Appendix

A.1 Baseline Algorithm Description

In our evaluations we show 3 additional unlearning algorithms alongside RELOAD. These algorithms are Gradient Ascent (GA), Finetuning (FT), and Gradient Ascent with Finetuning (GA FT).

Gradient Ascent [22] is a naive unlearning algorithm in which the forget dataset is optimized on, taking the negative of the typical loss function in order to perform gradient ascent instead of gradient descent.

Finetuning [24] utilises the concept of catastrophic forgetting [17] from the field of Continual Learning. By performing gradient descent optimisation on the retain set, this method aims to induce catastrophic forgetting of the forget set which is no longer being optimized over.

Gradient Ascent with Finetuning is an enhanced version of GA. The typical issue with GA is that it forgets well but destroys performance on the retain set, yielding an unlearned but poor performing model. Thus, this method adds additional steps of gradient descent optimisation on the retain set post the gradient ascent procedure to repair the damaged performance.

A.2 Reload Main Intuition and Justification

Before defining the steps of our algorithm, we discuss the motivating factors and justifications for this design. Existing evidence [19] [2] suggests that deep neural networks (DNNs), especially over-parameterized DNNs, tend to memorize parts of the training dataset. Thus it stands to reason that particular parameters in a neural network have learned disproportionately more about different parts of the training data, or have even memorized parts of it. The work by Xu et al. [25] even suggests that memorizing parts of the training set is a necessary component for DNNs to achieve optimal performance. Therefore, given a sufficiently trained model, parts of the training set are memorized in its parameters. Thus, if parameters that are particularly knowledgeable about or have memorized the difference between the $\mathcal{D}$ and $\mathcal{D}_{retain}$ can be identified, selectively retraining them should approximate a model that was trained with $\mathcal{D}_{retain}$ in the first place.

Consider a gradient descent update on a model $\mathcal{M}_\theta$, over some data x. This gradient points in the direction of steepest descent on the loss landscape, and taking a step in this direction moves θ towards the setting of parameters that allows $\mathcal{M}$ to perfectly fit the data x. Thus, intuitively, one can say that the parameters that are being updated the least (those with the lowest magnitude gradients) are those which know the most about x. We can measure this by considering the parameters with the lowest absolute gradients across x.

In the context of unlearning, we can extend this notion to consider the summed gradients over the entire forget set $\mathcal{D}_{forget}$. Consider the lower the magnitude of the parameter's gradient across $\mathcal{D}_{forget}$, implies the parameter knows more about $\mathcal{D}_{forget}$. This notion is motivated by gradient-based input saliency maps [20] [21] and the existing work on saliency unlearning by **?**]. We can extend this notion to $\mathcal{D}$, and by taking the ratio of these gradients, derive a measure for how much a parameter knows specific to the numerator dataset ($\mathcal{D}_{forget}$) that is not general to the denominator (entire) dataset $\mathcal{D}$. Letting $\nabla\mathcal{L}(\mathcal{D})$ denote the gradients with respect to $\mathcal{D}$, we can define this measure as so, which we call the knowledge-value:

$$\frac{|\nabla\mathcal{L}(\mathcal{D}_{forget})|+\epsilon}{|\nabla\mathcal{L}(\mathcal{D})|+\epsilon}=\frac{|\nabla\mathcal{L}(\mathcal{D})-\nabla\mathcal{L}(\mathcal{D}_{retain})|+\epsilon}{|\nabla\mathcal{L}(\mathcal{D})|+\epsilon} \tag{1}$$

In this equation, ϵ is a small value (around 1^{-10}) to prevent divide-by-zero situations and so that in the case where the numerator gradient is 0 and the denominator gradient is not, this ratio still measures relative importance to a dataset.

By the linearity of differentiation, since $\mathcal{D}_{retain}$ is $\mathcal{D}$ with $\mathcal{D}_{forget}$ removed, $\nabla\mathcal{L}(\mathcal{D}_{forget}) = \nabla\mathcal{L}(\mathcal{D}) - \nabla\mathcal{L}(\mathcal{D}_{retain})$. Thus access to $\mathcal{D}_{forget}$ in the unlearning case, is not required to compute this ratio, as long as the accumulated gradients on $\mathcal{D}$ are stored and available.

A.3 Justification of Non-Recoverability

Storing the gradients does not consitute a privacy breach in the softmax classification setting, because this choice does not not permit recovery of the instances within $\mathcal{D}_{forget}$ in this common setting.

Definition 1 (Recoverability). *Consider some data, $\mathcal{D} \in \mathscr{D}$, and consider a transformation $f : \mathscr{D} \to \mathcal{Q}$ that maps $\mathcal{D}$ into an arbitrary output space $\mathcal{Q}$. $\mathcal{D}$ is* recoverable *if f is injective.*

By the linearity of differentiation, we write $\nabla_\theta\mathcal{L}(\mathcal{D}) - \nabla_\theta\mathcal{L}(\mathcal{D}_{retain})$ as $\nabla_\theta\mathcal{L}(\mathcal{D}_{forget})$.

Note that from the gradients alone we cannot tell how many items were in $\mathcal{D}_{forget}$ when it was removed.

Then, if there was more than one item in $\mathcal{D}_{forget}$, then $\nabla_\theta \mathcal{L}(\mathcal{D}_{forget}) = \sum_{(X_i,Y_i)\in\mathcal{D}_{forget}} \nabla_{\theta_k}\mathcal{L}((X_i,Y_i),\hat{Y}_i)$. Given that this is a summation, and addition is not an injective operation - we cannot recover the gradients of the individual datapoints from this sum.

In the case that there was a single datapoint in $\mathcal{D}_{forget}$, then $\nabla_\theta \mathcal{L}(\mathcal{D}_{forget}) = \nabla_\theta \mathcal{L}((X_1,Y_1),\hat{Y}_1)$ where $\mathcal{D}_{forget} = \{(X_1,Y_1)\}$. Given this gradient, and the setting of cross-entropy loss and softmax activation, we know that the gradient expands to $-\nabla_{\theta_k}\sum_{i=1}^{C} Y_{1i}\log\hat{Y}_{1i} = \nabla_{\theta_k}\log\hat{Y}_{1j} = \frac{1}{\hat{Y}_{1j}}$ assuming Y is onehot encoded and this sample belongs to class j.

Then from this gradient we have obtained the j'th output of the model, which is $\hat{Y}_{1j} = \frac{e^{Z_{1j}}}{\sum_{i=1}^{C} e^{Z_{1i}}}$ where Z_1 are pre-softmax logits. For any element of Z_1, Z_{1k} of Z_1, $e^{Z_{1k}} = \hat{Y}_{1k}\cdot\sum_{i=1}^{C} e^{Z_{1i}}$ which cannot be calculated without knowing all of the other elements of Y_1. Thus, given a single output, none of Z_1 can be calculated, and thus the gradient cannot identify the input from which it was calculated.