Short Literature Review on Adaptive Machine **Un**learning Theory

Sushma Akoju
Project report, CSC 588 Spring 2024

April 2024

1 Introduction to Machine Unlearning

User's *right-to-be-forgotten* has been in discussion by many laws and regulations such as European Union's Article 17 of General Data Protection Regulation (GDPR). Since removing the data is not sufficient as Machine Learning models tend to memorize the training data that is malicious or unauthorized or sensitive. Such memorization poses risks in privacy breaches and security breaches as well as performance degradation. Machine Unlearning emerged as a solution space to remove influence of the specific training data points from the Machine learning models.

The Machine Unlearning challenge from Neurips 2023 [1] is based on [6] [10] [8]. where the definition of Machine Unlearning is: "unlearning refers to removing the influence of subset of training set from the weights of a trained model". In a database, we can issue a command on the terminal to access and performan changes. So it is relatively easy to issue a command of the form for any Database update/delete requests:

DELETE FROM USERS WHERE user_id $==$ $\boldsymbol{XYZ}$.
UPDATE USERS SET $RECORD_EXPIRES = today()$ **WHERE ....**
INSERT INTO USERS (col1, ...) VALUES (val1,...).

But in Deep Learning models, we do not have such flexibility to quickly delete as such models tend to memorize the data and corresponding influence of the data. So to acheive Machine Unlearning on a Deep Learning model, a traditional method would be to divide the data into training data and the forget set and retrain the model over training data without the forget set to acheive the results of "forgetfulness".

A common approach is *exact unlearning*, where in we train a model from the scratch with a new dataset without the forget set at all. Other method used is *approximate unlearning*, where machine learning model and dataset are modified to achieve approximately same effects as exact unlearning. Approximate unlearning appeals to the fact that without retraining the entire model, a new model without the user's data and the influence can be removed from the trained model. Let D be the data collected. Let $u \in U$, where U is the entire list of users available in the Dataset D. According to the GDPR, any user u has a right to request removal of data.

2 SISA [3] and Exact Machine Unlearning methods and Analysis

This section introduces Sharded, Isolated, Sliced, and Aggregated training (SISA) training framework used for Exact Machine Unlearning and explores a survey on Exact Machine Unlearning methods. Unleaning is challenging because:

1. We have limited understanding of the how each data point impacts the model. The study of influence of particular data point on the parameters of the model is limited. There is research that looks for test-time prediction to back to training, but it requires second-order derivatives of model's training algorithm.
2. There is stochasticity in training since we select small batches of data with 32 points that are randomly sampled dataset and ordering of such batches varies between different epochs. Training is parallelized without explicit synchronization as training is carried out in parallel threads with random ordering, that makes training non-deterministic.
3. Incremental Training also poses a restriction, i.e for any update at particular training point, all subsequent epochs and all subsequent model updates will depend on one such training point.
4. Stochasticity of learning i.e. PAC learning theory suggests learned hypothesis is one of the many hypothesis that minimize empirical risk. For example, a natural choice for optimizer for neural networks is stochastic gradient descent that can converge to one of many local minima. However there is a challenge to correlate the data point back to the hypothesis that was learned.

The need for differential privacy also suggests that the influence of the data point has bounds on the influence the point has on the model once the point is unlearned. This indicates the influence is non-zero. There is technically small, but non-zero influence on the model.

The Statistical Query Learning where model unlearning in statistical query learning framework such that it allows to unlearn the data point when the learning algorithm queries data in an order that was decided prior to training. This is easy to see how PAC Learning has corresponding SQ learning equivalent. But for complex algorithms is sufficiently challenging.

2.0.1 Differential Privacy

Differential privacy is an approach that provides privacy when sharing information about group of individuals by witholding information about the individuals. Let ϵ be a positive real number and $\mathcal{A}$ is a randomized algorithm. Let im $\mathcal{A}$ be the image of $\mathcal{A}$. $\mathcal{A}$ is said to provide ε-differential privacy if, for all datasets D_1, D_2 that differ on single point (i.e. data of one user) and $\mathcal{S}$ all subsets in im-$\mathcal{A}$ then,

$$\frac{Pr[\mathcal{A}(D_1) \in \mathcal{S}]}{Pr[\mathcal{A}(D_2) \in \mathcal{S}]} \leq e^{\varepsilon}$$

This indicates that the computations of $\mathcal{A}$ on two different datasets D_1, D_2 remains statistically indistinguishable.

2.0.2 Machine Unlearning

For Machine Unlearning, the two computations (sequence of deletes vs full retraining) run on same dataset D remains statistically indistinguishable.

2.0.3 Sharded, Isolated, Sliced, and Aggregated training (SISA) [3]

This section is a summary and my analysis of [3] paper is a required reading for [5] since Adaptive Machine Unlearning is dependent on SISA [3].

The idea here is that this method replicates model being trained for the dataset D, $|D| = N$ that is partitioned into training data D_{tr} and test data D_{te}. The D_{tr} is further partitioned into some S disjoint shards. Each shard is split into K slices. Each disjoint shard $D_s, s \in \{1, \cdots S\}$ is given each one of the replica models M_s. The constituent models have no flow of information between them. Slicing is possible for any iterative learning algorithm that is stateful. Gradient descent seems to right candidate for this requirement.

The requirement of training process is that updates obtained during iterative training are not exchanged between constituent models.

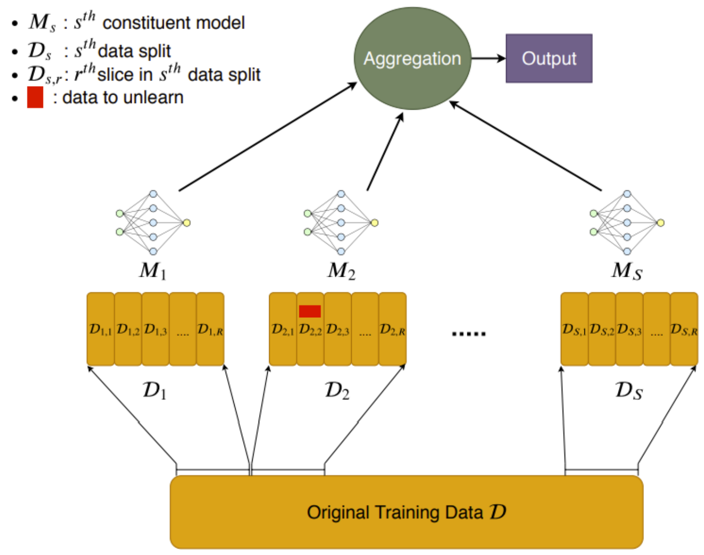
• M_s : s^{th} constituent model
• $\mathcal{D}_s$: s^{th} data split
• $\mathcal{D}_{s,r}$: r^{th} slice in s^{th} data split
• 🟥 : data to unlearn
Aggregation
Output
M_1
M_2
M_S
$\mathcal{D}_{1,1}$ $\mathcal{D}_{1,2}$ $\mathcal{D}_{1,3}$ .... $\mathcal{D}_{1,R}$
$\mathcal{D}_{2,1}$ $\mathcal{D}_{2,2}$ $\mathcal{D}_{2,3}$ .... $\mathcal{D}_{2,R}$
$\mathcal{D}_{S,1}$ $\mathcal{D}_{S,2}$ $\mathcal{D}_{S,3}$ .... $\mathcal{D}_{S,R}$
.....
$\mathcal{D}_1$
$\mathcal{D}_2$
$\mathcal{D}_S$
Original Training Data $\mathcal{D}$

Fig. 2: **SISA** training: data is divided in shards, which are themselves divided into slices. One constituent model is trained on each shard by presenting it with incrementally many slices and saving its parameters before the training set is augmented with a new slice. When data needs to be unlearned, only one of the constituent models whose shards contains the point to be unlearned needs to be retrained — retraining can start from the last parameter values saved before including the slice containing the data point to be unlearned.

Figure 1: SISA [3]

1. **Sharding:** A key assumption is that there is no prior information about distribution of unlearning requests i.e. no information about probabilities with which each data point might be unlearned. The dataset D is uniformly partitioned into S shards such that $\cap_{k\in|S|}D_k = \phi$ and $\cup_{k\in|S|}D_k = D$. M_k is the corresponding model that is trained over the shard D_k. Let user u's data point be d_u. If user requests for unlearning d_u, unlearning is carried out in two steps

   a Locate the dataset, and the shard in which d_u is present and let that be D_u.

   b Retrain the corresponding model M_u, from the scratch on $D_u \backslash d_u$ so resulting model is M'_u. For comparison $|D| \backslash d_u >> D_u \backslash d_u$ so time to retrain a baseline M seems to be far greater than retraining M_k.

2. **Isolation:** There is degradation of generalization in the aggregated model from isolated training. Appropriate choice of shards could help improve

the generalization.

3. **Isolation:** Model training is carried out incrementally i.e. R disjoint slices $\forall i \in \{1 \cdots |R|\}, D_{k,i}$, train $M_{k,0}$ with e_1 epochs and $D_{k,0}$. Train $M_{k,0}$ with $D_{k,1}$ dataset so resulting model is $M_{k,1}$. $M_{k,R-1}$ is trained with $D_{k,R}$, e_R epochs so the resulting model is $M_{k,R} = M_k$. If user requests for unlearning d_u, unlearning is carried out in two steps

   a Locate the dataset, and the shard in which d_u is present and let that be D_u.

   b Retrain the corresponding model M_u, from the scratch on $D_u \backslash d_u$ so resulting model is M'_u. For comparison $|D| \backslash d_u >> D_u \backslash d_u$ so time to retrain a baseline M seems to be far greater than retraining M_k. N is size of the dataset , $D = \frac{N}{S}$ where S is number of shards. e' is number of epochs without slicing, then $e = \sum_{i]1}^{R} e_i$ where R is number of slices. e_i is number of epochs required to train $\frac{iD}{R}$ samples. $e' = \sum e_i \frac{iD}{R} \equiv e = \frac{2R}{R+1} * e'$

4. **Aggregation:** The goals of aggregation are to maximize the joint predictive performance of constituent models. This is based on partitioning of Sharding. And aggregation should not involve training data. Since there is no prior information about which data unlearning requests can occur, so data is partitioned uniformly and later by using label based majority voting.

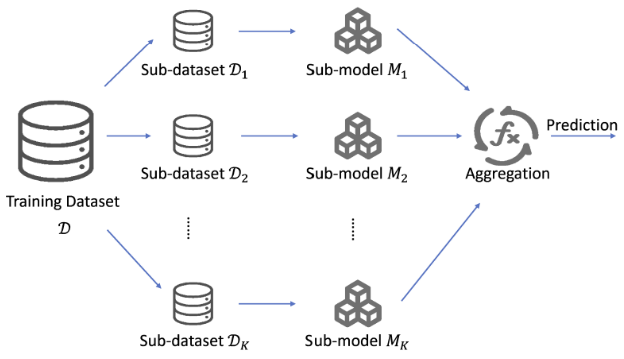

Fig. 2: SISA Framework

Figure 2: SISA [9]

2.0.4 Challenges

Weak learners are not effective using SISA approach. The complex tasks such as few-shot over MNIST do suffer, since M_k could be weak if shards are smaller.

For example, in Imagenet, there are 150,000 features and 1000 classes, and require order of $>>$ 100 hidden layers. So naturally smaller shards introduce overfitting due to number of samples per class.

Hyperparameter search is another challenge. Training is assumed to be $O(SR)$ but it seems to eventually reduce to $O(R)$.
The time related guarantees seem reasonable, so far number shards are not more than the size of dataset. Benefits of sharding are noticeable when $K << N$.

The worst case scenario where there are K number of unlearning requests, requiring to conduct unlearning over each one of the K M_k models which gives no speedup. This is attributed to number of samples per class.

The studies over SISA observed sharding induces accuracy degrading. SISA also requires additional storage to retain the model and datapoint's influence within each slice. [9] SISA generally seems to struggle with highly dependent data while it effectively improves unlearning efficiency at the risk of accuracy degradation and additional storage costs.

SISA does explore distribution aware sharding and adaptable unlearning. However, the framework explored breadth of the topic than the depth. The questions remain: how do the weights and aggregated performance gets impacted under certain case scenarios. Can we conduct test-time prediction back to training after an unlearning request was implemented? (to asses the true performance in terms of losses, impact on the learnt derivatives).

While differential privacy addresses increasing privacy needs and demands but what would happen if most users want some datapoint or the other removed frequently?? Can there be an intersection of differential privacy and Machine unlearning methods that can adapt to time-based data removal?

Some datapoints only contribute to accuracy until some time, for example, we might be less likely to look at Covid data - photos, images, videos in 2025. Similarly war related news, images and videos also has time based constraints. For example we do not have an option to unsee the data on the social networks esp. even if the images were that of one person who is directly impacted by war. So any user can request deletion, which means multiple deletion/update requests for unlearning might occur. **So I think Time-based as well as majority-voted Unlearning requests may provide insights into having some idea about distributions of underlying unlearning requests and likelihood of receiving unlearning requests.**

2.0.5 Exact Machine Learning algorithms [9]

Several unlearning frameworks are reviewed in [9]. This work provides great insights distinguishing between supervised models, clustering approaches, divide and conquer methods etc.

The paper also introduces what is Approximate Machine Unlearning. The methods involves computation of influence and adjustment of model parameters, addition of noise and validation of updated model.

$\mathcal{F}(\mathcal{D}, w) = \sum_{z \in \mathcal{D}} f(z; w) + \frac{\lambda n}{2} ||w||_2^2$ where f is the convex loss. To protect the information pertaining to removed data points, [4] propose to add random perturbation to loss function during training process to protect the gradient information.

The influence of $z' = (x', y')$ on original model, that is to be removed, is $-H_{w^*}^{-1}\Delta$, where $\Delta = \lambda w^* + \nabla f(z'; w)$ and $-H_{w^*}^{-1} = \nabla F(D_r; w), D_r = D \backslash z'$. Adjust model parameters w^*, $w^- = w^* - (-H_{w^*}^{-1}\Delta) = w^* + H_{w^*}^{-1}\Delta$.
This process of removal of data by adding random peturbation to protect leakage of information is certified-removal of information.
And the [9] discusses several approaches for Approximate Unlearning based on Influence Function.

To overcome the computation costs for Hessian matrix in [4], use of Fisher Information Matrix (FIM) was recommended. There was another interesting approach Learning with Selective Forgetting (LSF) that partitions loss into 4 : classification loss, mnemonic loss (tying class to corresponding embedded code), selective forgetting loss (to select classes for removal), and regularization loss to prevent catastrophic forgetting.

One common observation among all frameworks including SISA, is that they are not thoroughly run after an unlearning request. There is need to compare the performance between for example, Imagenet versus SISA implementation after atleast 1 unlearning request. This makes way for why we need formal definition of Machine Unlearning and Adaptive Machine Unlearning.

3 Adaptive Machine Unlearning [5]

Adaptive Machine Unlearning introduced in [5] uses SISA framework.
The general goal of Machine unlearning is that the two computations (sequence of deletes vs full retraining) run on same dataset D remains statistically indistinguishable.

3.1 (α, β)- Unlearning, Exact & Approximate Unlearning

DEFINITION: Exact unlearning. R_A is an unlearning algorithm for A for all datasets D, and all deletion sequences $\{Z_1, z_2 \cdots z_i\}$, following condition holds, for every deletion step t: $\hat{\theta}_t \underset{d}{=} \hat{\theta}'_t$, and $\hat{\theta}$ is the model that is updated or retrained.

DEFINITION: Approximate unlearning. We say, R_A is an (α, β)-unlearning algorithm for A, if for all datasets D, and all deletion sequences $\{Z_1, z_2 \cdots z_i\}$, following condition holds: for every deletion step t, $\forall E, Pr[\hat{\theta}_t \in E] \leq e^{\alpha}.Pr[\hat{\theta}'_t \in E] + \beta$.

$\hat{\theta}'_t$ are output models retrained using A and $\hat{\theta}_t$ models output by unlearning algorithm R_A. α, β are stronger unlearning guarantees.

Differential Privacy (DP) compares same algorithm over different datasets. Data deletion compares different algorithms run on same dataset.

3.1.1 Deletion in Convex models

1. Convex loss function: $l : \mathbb{R}^d \times Z \to \mathbb{R}$
2. Dataset $D = \{z_1, z_2, \cdots z_n\} \in Z^n$.
3. Want to solve ERM problem $\min\limits_{\theta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1} l(\theta, z_i) := L(\theta, D)$
4. Gradient Descent Initialization $\theta_t = \theta_{t-1} - \eta \nabla L(\theta_{t-1}, D)$.
5. Accuracy $L(\theta, D) - \min\limits_{\theta} L(\theta, D)$
6. Deletions are not i.i.d

If loss function is only weak convex, regularize. We would not reach exact OPT, so path dependence need not be a concern. During training phase, run distributed Gradient descent until ϵ-convergence is guaranteed. Output final model perturbed with zero-mean Gaussian noise of scale $\approx \epsilon.\sqrt{log(1/beta)}/\alpha$.

During unlearning phase,

1. For $t = 0, 1, 2 \cdots$
2. Input: deletion request for z_t
3. Remove z_t from the dataset
4. Run distributed Gradient descent, initialized at output of previous round, until ϵ-convergence is guaranteed.
5. Output final model perturbed with zero-mean Gaussian noise of scale $\approx \epsilon.\sqrt{log(1/beta)}/\alpha$.

3.1.2 (α, β, γ)-unlearning

(Update Requester (UpdReq): The update sequence is generated by an update requester which is modeled by a (possibly randomized) mapping $UpdReq$ $\psi^* \times (Z \times T) \to Z \times T$, this takes as input the history of herself and other algorithms and outputs a new update for current round. For update request sequences $U = \{u^t\}_t$ $u^0 = UpdReq(\psi^0), u^1 = UpdReq(\psi^0, u^0, \psi^1), \cdots u^t = UpdReq(\psi^0, u^0, \cdots \psi^{t-1})$.

Update requests are non-adpative if the objects are independent if $u^t = UpdReq(\psi^0, u^0, \cdots \psi^{t-1}) = UpdReq(u^0, u^1 \cdots u^{t-1})$

DEFINITION: (α, β, γ)**-unlearning.** We say, R_A is an (α, β, γ)-unlearning algorithm for A, if for all datasets $D = D^0$ and all update requesters $UpdReq$, following condition holds: for every update step t,
$\forall E \subseteq \Theta^* : Pr[R_A(D^{t-1}, u^t, s^{t-1}) \in E | u^{\leq t}] \leq e^{\alpha} pr[A(D^t) \in E] + \beta$.

R_A is nonadapative if (α, β, γ)-unlearning reduces to nonadaptive.

$\hat{\theta}'_t$ are output models retrained using A and $\hat{\theta}_t$ models output by unlearning algorithm R_A. α, β are stronger unlearning guarantees.

3.1.3 Theorem

The theorem 3.1 is general adaptive unlearning, is reducible to nonadaptive over distributed learning algorithms. The joint distribution of update requests and internal states are not changed. So r is redrawn after each update from its conditional distribution conditioned on $u^{\leq t}$. Since publishing function is differentially private in r by lemma 2.1 post processing preserves differential privacy as well as update sequence. The max-information bound is applied (theorem 2.1) which shows in terms of conditional distribution in r to its original prior distribution P^m. Resampling r from P^m removes dependence between r and Update sequences so it reduces to non-adaptive case and allows to apply hypothesized unlearning guarantees for nonadaptive update requesters.

Intuitively, under distributed learning algorithm setting, A^{distr} (where A^{single} is single shard learning algorithm which selects sampler that selects points in a shard) the $R_{A^{distr}}$ is a non-adaptive $(0, 0, 0)$-unlearning algorithm for A^{distr}. This is theorem 4.1. Combining theorems 4.1 and 3.1, unlearning guarantees are $\alpha = O(\epsilon^2 k + k\sqrt{\delta/\epsilon}), \beta = \gamma = O(\sqrt{e^{-\epsilon^2 k} + k\sqrt{\delta/\epsilon}})$.

The runtime guarantees assume DP requriements are staisfied. This generally signifies that every shard that contains deletion point needs to be retrained. For nonadpative case, only one shard needs to be retrained in expectation and a high probability bound using Hoeffding's inequality.

The empirical tests on datasets, shows under SISA framework, failure of satisiying adaptive deletion requests occurs and DP can mitigate the same.

The theorem seem to achieve desired properties and unlearning guarantees based on few empirical tests.

The tests on null hypothesis: fail to reject the null hypothesis that the algorithm has adaptive data deletion guarantees at $p \leq 0.05$. The tests show their (α, β, γ)-unlearning approach on a Adaptive learning algorithm performs better than SISA.

3.1.4 Theorem 3.1 proof

The theorem proof is as follows, in simple steps:

1. (ϵ, δ)-differentially private in $r \approx p^m$.

ϵ is the max distance between query on database of D_1 and query database of D_2, where D_1 and D_2 are different datasets differ by one data point. ϵ is a metric of privacy loss under a differential change. Smaller ϵ yeilds better privacy since outputs are more alike but less accurate. when $\epsilon = 0$, outputs from all databases for a given query are same.

δ is a probability of information accidentally leaked. This is directly proportional to size of database. As the size of database increases, so does the δ likelihood of data being leaked.

Privacy loss will be bounded by ϵ with probability of $1 - \delta$.

2. Fixed pair of unlearning and learning algorithms (R_A, A).
3. Applying max-information bounds. Uses probability by joint distribution over $r, u^{\leq t}$. If R_A is non-adaptive, then $\exists \alpha', \beta', \gamma'$ each one expressed in terms of ϵ' and δ'. So it is easy to see Good event bound is $\sqrt{\delta'}$ so the updates generate differently-private .
4. with probability of $1 - \delta'$ over the draw of $u^{\leq t}$ for every event F in space of random seeds, r, an Good event bounded guarantee is obtained. This seems to ensure bounded on probability of information accidentally leaked as well as such that distance between queries is maintained to ensure adaptive updates.

   The proof shows that probability that an R_A belongs to set $E \subseteq \Theta^*$ given update sequences has to bear a cost of $\sqrt{\delta'}$ that it is *not* a good event (i.e. likelihood of leakage determines the penalty of finding such an R_A that allows for distance over queries on different models is bounded to be exponential terms in terms of α', ϵ' is a factor of probability that $A \in E$.

   Shards are independently selected. So but I assume that the selection of shards seems also to depend on the DP-aware selection to some extent.

   I had a couple of questions, does this mean that Algorithm A needs to learn such that it is DP-aware?

3.2 About naming convention

Machine Unlearning naming convention seems to be less intuitive. There was a debate among neuroscientists and cognitive science researchers about the distinction between machine learning and human unlearning. Human unlearning [2] seems helpful in forming the hypothesis during the decisive learning process, by making the learning less decisive. On another note, metacognition studies [7] suggest people's engagement with corrections to errors that were made with high confidence are more likely to be learned and corrected. This underpins the process unlearning to correct an answer. The word Machine unlearning was coined by Yinzhi Cao of Lehigh University and Junfeng Yang of

Columbia University `https://tinyurl.com/munlearning-history`. However, since we consider terms such as remember or forget or change some data sample - all of which refer to the existence of such an instance during the learning process i.e. during the training phase. It might be better to come up with a new suitable naming convention. I would be in favor of Machine Relearning since the model does not only need to remove a data sample but its influence over the weights learned by the trained model. I believe models never really unlearn as traces of *learned* information and/or the influence of trained data points continue to persist due to the limitations of time and computation.

3.3 Conclusion

Machine Unlearning is achieved differently based on the type of tasks such as simple vs complex tasks, as discussed in the earlier section. Since the advent of Language Models and ImageNet, Inception so on, the zero-shot and few-shot learning has become more popular in terms of transfer learning. With the introduction of Large Language Models such as GPTx, LLAMA, Claude, and so on, there is limited scope and access to these models. However, there are different methods explored. The theoretical work within the Machine Unlearning space has significant relevance to its applications over more advanced and restricted Deep Learning models.

References

[1] NeurIPS 2023. Neurips 2023 machine unlearning challenge, 2023. `https://unlearning-challenge.github.io/` [Accessed: (September 2023)].

[2] Ganesh Baliga, John Case, Wolfgang Merkle, Frank Stephan, and Rolf Wiehagen. When unlearning helps. *Information and Computation*, 206(5):694–709, 2008. Special Issue: The 17th International Conference on Concurrency Theory (CONCUR 2006).

[3] Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning, 2020.

[4] Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. Certified data removal from machine learning models. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3832–3842. PMLR, 13–18 Jul 2020.

[5] Varun Gupta, Christopher Jung, Seth Neel, Aaron Roth, Saeed Sharifi-Malvajerdi, and Chris Waites. Adaptive machine unlearning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 16319–16330. Curran Associates, Inc., 2021.

[6] Alessandro Mantelero. The eu proposal for a general data protection regulation and the roots of the 'right to be forgotten'. *Computer Law & Security Review*, 29(3):229–235, 2013.

[7] Janet Metcalfe, Matti Vuorre, Emily Towner, and Teal S Eich. Curiosity: The effects of feedback and confidence on the desire to know. *Journal of Experimental Psychology: General*, 2022.

[8] Thanh Tam Nguyen, Thanh Trung Huynh, Phi Le Nguyen, Alan Wee-Chung Liew, Hongzhi Yin, and Quoc Viet Hung Nguyen. A survey of machine unlearning. *arXiv preprint arXiv:2209.02299*, 2022.

[9] Jie Xu, Zihan Wu, Cong Wang, and Xiaohua Jia. Machine unlearning: Solutions and challenges. *IEEE Transactions on Emerging Topics in Computational Intelligence*, page 1–19, 2024.

[10] Haibo Zhang, Toru Nakamura, Takamasa Isohara, and Kouichi Sakurai. A review on machine unlearning. *SN Computer Science*, 4(4):337, 2023.