Fast Yet Effective Machine Unlearning

Ayush K. Tarun, Vikram S. Chundawat, Murari Mandal, and Mohan Kankanhalli, *Fellow, IEEE*

***Abstract*— Unlearning the data observed during the training of a machine learning (ML) model is an important task that can play a pivotal role in fortifying the privacy and security of ML-based applications. This article raises the following questions: 1) can we unlearn a single or multiple class(es) of data from an ML model without looking at the full training data even once? and 2) can we make the process of unlearning fast and scalable to large datasets, and generalize it to different deep networks? We introduce a novel machine unlearning framework with error-maximizing noise generation and impair-repair based weight manipulation that offers an efficient solution to the above questions. An error-maximizing noise matrix is learned for the class to be unlearned using the original model. The noise matrix is used to manipulate the model weights to unlearn the targeted class of data. We introduce impair and repair steps for a controlled manipulation of the network weights. In the impair step, the noise matrix along with a very high learning rate is used to induce sharp unlearning in the model. Thereafter, the repair step is used to regain the overall performance. With very few update steps, we show excellent unlearning while substantially retaining the overall model accuracy. Unlearning multiple classes requires a similar number of update steps as for a single class, making our approach scalable to large problems. Our method is quite efficient in comparison to the existing methods, works for multiclass unlearning, does not put any constraints on the original optimization mechanism or network design, and works well in both small and large-scale vision tasks. This work is an important step toward fast and easy implementation of unlearning in deep networks. Source code: https://github.com/vikram2000b/Fast-Machine-Unlearning.**

***Index Terms*— Data privacy, forgetting, machine unlearning, privacy in artificial intelligence (AI).**

I. Introduction

CONSIDER a scenario where it is desired that the information pertaining to the data belonging to a single class or multiple classes be removed from the already trained machine learning (ML) model. For example, a company is requested to remove the face image data for a user (or a set of users) from the already trained face recognition model. In addition, there is a constraint such that the company no longer has access to those (requested to be removed) facial images. How do we solve such a problem? With the increase in privacy awareness among the general populace and the cognizance of the negative impacts of sharing one's data with ML-based applications, such type of demands could be raised frequently in near future. Privacy regulations [1], [2] are increasingly likely to include such provisions in future to give the control of personal privacy to the individuals. For example, the California Consumer Privacy Act (CCPA) [2] allows companies to collect user data by default. However, the user has the *right to delete* her personal data and *right to opt-out* of the sale of her personal information. In case a company has already used the data collected from the users (in our example, *face data*) to train its ML model, then the model needs to be manipulated suitably to reflect the data deletion request. The naive way is to redo the model training from scratch for every such request. This would result in significant cost of time and resources to the company. How can this process be made more efficient? What are the challenges? How do we know that the model has actually unlearned those class/classes of data? How to ensure minimal effect on the overall accuracy of the model? These are some of the questions that have been asked and possible solutions have been explored in recent times [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19].

The unlearning (also called selective forgetting, data deletion, or scrubbing) solutions presented in the literature are focused on simple learning algorithms such a linear/logistic regression [20], random forests [7], and k-means clustering [11]. Initial work on forgetting in convolutional networks is presented in [9] and [10]. However, these methods are shown to be effective only on small scale problems and are computationally expensive. Efficient unlearning in deep networks such as convolutional neural networks (CNNs) and vision transformers (ViTs) still remain an open problem. In particular, efficiently unlearning of multiple classes is yet to be explored. This is due to several complexities that arise while working with deep learning models. For example, the nonconvex loss space [21] of CNNs makes it difficult to assess the effect of a data sample on the optimization trajectory and the final network weight combination. Furthermore, several optimal set of weights may exist for the same network, making it difficult to confidently evaluate the degree of unlearning. Comparing the updated model weights after unlearning with a model trained without the forget classes might not reveal helpful information on the quality of unlearning. Forgetting a cohort of data or an entire class of data while preserving the accuracy

Manuscript received 10 July 2022; revised 12 January 2023 and 3 March 2023; accepted 4 April 2023. Date of publication 1 May 2023; date of current version 4 September 2024. This work was supported by the National Research Foundation, Singapore under its Strategic Capability Research Centers Funding Initiative. *(Ayush K. Tarun and Vikram S. Chundawat contributed equally to this work.) (Corresponding author: Murari Mandal.)*

Ayush K. Tarun and Vikram S. Chundawat are with the Mavvex Labs, Faridabad 121001, India (e-mail: ayushtarun210@gmail.com; vikram2000b@gmail.com).

Murari Mandal was with the School of Computing, National University of Singapore, Singapore 117417. He is now with the School of Computer Engineering, Kalinga Institute of Industrial Technology (KIIT), Bhubaneswar 751024, India (e-mail: murari.mandalfcs@kiit.ac.in).

Mohan Kankanhalli is with the School of Computing, National University of Singapore (NUS), Singapore 117417 (e-mail: mohan@comp.nus.edu.sg).

This article has supplementary downloadable material available at https://doi.org/10.1109/TNNLS.2023.3266233, provided by the authors.

Digital Object Identifier 10.1109/TNNLS.2023.3266233

of the model is a nontrivial problem as has been shown in the existing works [3], [9]. Moreover, efficiently manipulating the network weights without using the unlearning data still remains an unsolved problem. Other challenges are to unlearn multiple classes of data, perform unlearning for large-scale problems, and generalize the solution to different types of deep networks.

Estimating the effect of a data sample or a class of data samples on the deep model parameters is a challenging problem [22], [23]. Therefore, several unlearning research efforts have been focused on the simpler convex learning problems (i.e., linear/logistic regression) that offer better theoretical analysis. Researchers have [22] used influence functions to study the behavior of black-box models such as CNNs through the lens of training samples. It is observed that data points with high training loss are more influential for the model parameters. The adversarial versions [24] of the training images are generated by maximizing the loss on these images. It is further shown that the influence functions are also useful for studying the effect of a group of data points [23]. Recently, Huang et al. [25] proposed to learn an error-minimizing noise to make training examples unlearnable for deep learning models. The idea is to add such noise to the image samples that fools the model in believing nothing is to be learned from those samples. If used in training, such images have no effect on the model.

Unlearning requires the model to forget specific class(es) of data but remember the rest of the data. For the class(es) to be forgotten, if the model can be updated by observing patterns that are somehow *opposites* of the patterns learned at the time of original training, then the updated model weights might reflect the desired unlearning. And hopefully it preserves the remaining classes information. We know that the original model is trained by minimizing the loss for all the classes. So intuitively, maximizing a noise with respect to the model loss only for the unlearning class will help us learn such patterns that help forgetting. It can also be viewed as learning anti-samples for a given class and use these anti-samples to damage the previously learned information. In this article, we propose a framework for unlearning in a *zero-glance* privacy setting, i.e., the model can not see the unlearning class of data. We learn an error-maximizing noise matrix consisting of highly influential points corresponding to the unlearning class. After that, we train the model using the noise matrix to update the network weights. We introduce unlearning by selective impair and repair (UNSIR), a *single-pass* method to unlearn single/multiple classes of data in a deep model without requiring access to the data samples of the requested set of unlearning classes. Our method can be directly applied on the already trained deep model to make it forget the information about the requested class of data - while at the same time retaining very close to the original accuracy of the model on the remaining tasks. In fact our method performs exceedingly well in both unlearning the requested classes and retaining the accuracy on the remaining classes. To the best of our knowledge, this is the first method to achieve efficient multiclass unlearning in deep networks not only for small-scale problems (ten classes) but also for large-scale vision problems (100 classes). Our method works with the stringent *zero-glance* setting where *data samples of the requested unlearning class is either not available or can not be used*. This makes our solution unique and practical for real-world application. An important and realistic use-case of unlearning is face recognition. We show that our method can effectively make a trained model forget a single as well as multiple faces in a highly efficient manner, without glancing at the samples of the unlearning faces.

To summarize, our key contributions are as follows.

1) We introduce the problem of unlearning in a *zero-glance* setting which is a stricter formalization compared to the existing settings and offers a prospect for higher-level of privacy guarantees.
2) We learn an error-maximizing noise for the respective unlearning classes. UNSIR is proposed to perform single-pass impair and single-pass repair by using a very high learning rate. The impair step makes the network forget the unlearning data classes. The repair step stabilizes the network weights to better remember the remaining tasks. The combination of both the steps allows it to obtain excellent unlearning and retain accuracy.
3) We show that along with a better privacy setting and offering multiclass unlearning, our method is also highly efficient. The multiclass unlearning is performed in a single impair-repair pass instead of sequentially unlearning individual classes.
4) The proposed method works on large-scale vision datasets with strong performance on different types of deep networks such as convolutional networks and ViTs. Our method does not require any prior information related to process of original model training and it is easily applicable to a wide class of deep networks. Specifically, we show excellent unlearning results on face recognition. To the best of our knowledge, it is the first machine unlearning method to demonstrate all the above characteristics together.

II. Related Work

A. Machine Unlearning

Machine unlearning was formulated as a data forgetting algorithm in statistical query learning [26]. Brophy and Lowd [7] introduced a variant of random forests that supports data forgetting with minimal retraining. Data deletion in k-means clustering has been studied in [11] and [27]. Guo et al. [12] give a certified information removal framework based on Newton's update removal mechanism for convex learning problems. The data removal is certified using a variation of the differential privacy condition [28], [29]. Izzo et al. [14] presents a projective residual update method to delete data points from linear models. A method to hide the class information from the output logits is presented in [13]. This however, does not remove the information present in the network weights. Unlearning in a Bayesian setting using variational inference is explored for regression and Gaussian processes in [8]. Neel et al. [15] study the results of gradient descent based approach to unlearning in convex models. All

these methods are designed for convex problems, whereas we aim to present an unlearning solution for deep learning models.

Some methods adopt strategic grouping of data in the training procedure and thus enable smooth unlearning by limiting the influence of data points on model learning [4], [16]. This approach results in high storage cost as it mandates storing multiple snapshots of the network and gradients to ensure good unlearning performance. These approaches are independent of the types of learning algorithms and rely on the efficient division of training data. They also need to retrain a subset of the models, while we aim to create a highly efficient unlearning algorithm without any memory overhead. Gupta et al. [30] proposed an algorithm to handle a sequence of adaptive deletion requests in this setting.

B. Unlearning in Deep Neural Networks

Forgetting in deep neural networks is challenging due to their highly nonconvex loss functions. Although the term *forgetting* is used quite often in continual learning literature [31], where a model rapidly loses accuracy on the previous task when fine-tuned for a new task. This however does not address the information remaining in the network weights. Throughout this article we use the term *unlearning* and *forgetting* interchangeably, both denoting that the information of data in the network weights are also removed. Golatkar et al. [9] proposed an information theoretic method to scrub the information from intermediate layers of deep networks trained with stochastic gradient descent (SGD). They also give an upper-bound on the amount of remaining information in the network [32] after forgetting by exploiting the stability of SGD. This work is extended [10] by including an update mechanism for the final activations of the model. They present a neural tangent kernel (NTK) based approximation of the training process and use it to estimate the updated network weights after forgetting. However, both the approximation accuracy and computational costs degrade for larger datasets. The computational cost even in a small dataset is quite high as the cost is quadratic in the number of samples. Golatkar et al. [3], directly train a linearized network and use it for forgetting. They train two separate networks: the core model, and a mixed-linear model. The mixed-linear model requires Jacobian-vector product (JVP) computation and few other fine-tuning. This framework was shown to be scalable for several standard vision datasets. However, they present such a network only for ResNet50 which requires a lot of fine-tuning to obtain the results. Also designing a mixed-linear network for every deep architecture is an inefficient approach. Some researchers have studied the unintended privacy risks resulting from the existing unlearning methods [33], [34]. Thudi et al. [35] show the difficulty of formally proving the absence of certain data points in the model. They suggest that the current unlearning methods are well-defined only at the algorithmic level. Forgetting in federated learning [36] and recommendation systems [37] are also explored. Several other notable works include [38], [39], [40]. Our method does not put any constraints on the type of optimization to be used while training. We do not train any additional network, in-fact we do not require any prior information related to the training process. In addition, we propose the first Unlearning method that works for both CNN and ViTs. We show the results on different deep learning models, small and large datasets, and demonstrate successful unlearning in face-recognition.

C. Data Privacy

Privacy in ML has been extensively studied and various privacy-preserving mechanisms have been presented [41], [42]. The most common assumption in the such privacy protecting frameworks is that the model can freely access the entire training data and algorithms are devised to protect the model from leaking information about the training data. Another privacy setting [25], [43] considers a scenario where the goal is to make the personal data completely unusable for unauthorized deep learning models. The solutions in such a setting are based on the principles of the adversarial attack and defense methods [44], [45]. Some privacy settings [3], [14] allow the user to make a request to forget their data from the already trained model. These privacy settings assume having access to all the training data before forgetting. We propose to work in a stricter setting where once the user has made a request for forgetting her data (for example, her face in the face recognition model), the model can not use those samples even for the purpose of network weight manipulation.

III. Unlearning in Zero-Glance Privacy Setting

A. Zero-Glance Privacy Assumptions

In several use cases, the ML model is trained with facial images and personal medical data. Due to the sensitive nature of the data and the time constraints usually set by the data protection regulations (general data protection regulation (GDPR), CCPA), it may not be possible to use the forget set data even for unlearning purpose. We assume that the user can request for immediate deletion of her data and a time-bound removal of the information (in network weights) from the already trained model. The immediate removal of requested data leaves us with only the remaining data to perform unlearning. Once the network weights are updated, the model should not have any information corresponding to the forgetting data. Even after being exposed to the forgetting samples, the relearn time (RT) should be substantially high to ensure that the model has actually forgotten those samples.

B. Preliminaries and Objective

We formulate the unlearning problem in the context of deep networks. Let the complete training dataset consisting of n samples and K total number of classes be $\mathcal{D}_c = \{(x_i, y_i)\}_{i=1}^{n}$ where $x \in \mathcal{X} \subset \mathbb{R}^d$ are the inputs and $y \in \mathcal{Y} = 1, \ldots, K$ are the corresponding class labels. If the forget and retain classes are denoted by $\mathcal{Y}_f$ and $\mathcal{Y}_r$ then $\mathcal{D}_f \cup \mathcal{D}_r = \mathcal{D}_c$, $\mathcal{D}_f \cap \mathcal{D}_r = \emptyset$. Let the deep learning model be represented by the function $f_\theta(x) : \mathcal{X} \to \mathcal{Y}$ parameterized by $\theta \in \mathbb{R}^d$ used to model the relation $\mathcal{X} \to \mathcal{Y}$. The weights θ of the original trained deep network f_θ[1] are a function of the complete

[1]We use the notation f to denote the model in the rest of this article.

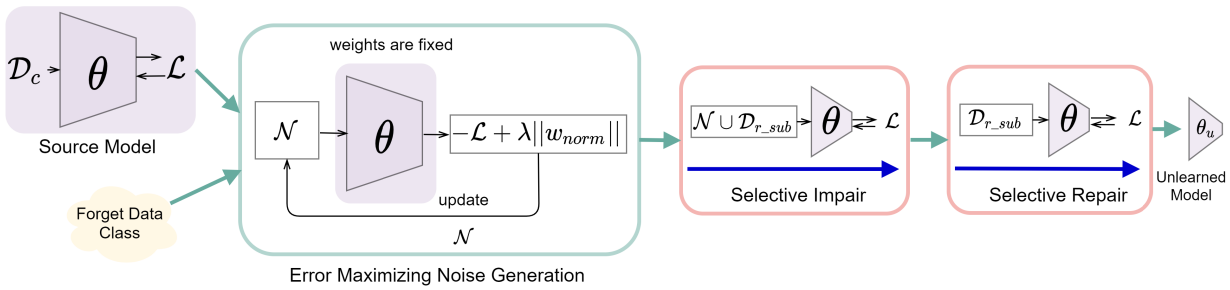

Fig. 1. Proposed unlearning framework. We use the pretrained model to learn the error-maximizing noise matrix for the unlearning class. The generated noise $\mathcal{N}$ is then used along with a subset of the retain data $\mathcal{D}_{r\_\text{sub}}$ to update the model with one epoch (impair). Next, we apply a healing step by further updating the network with only the retain data $\mathcal{D}_{r\_\text{sub}}$ (repair). The repair step helps in regaining the overall model performance while unlearning the requested class/classes of data.

training data $\mathcal{D}_c$. Forgetting in zero-glance privacy setting is an algorithm, which gives a new set of weights $\theta_{D_{r\_\text{sub}}}$ by using the trained model f and a subset of retain images $\mathcal{D}_{r\_\text{sub}} \subset \mathcal{D}_r$ which does not remember the information regarding D_f and behaves similar to a model which has never seen D_f in the parameter and output space.

To achieve unlearning, we first learn a noise matrix $\mathcal{N}$ for each class in $\mathcal{Y}_f$ by using the trained model. Then we transform the model in such a way that it fails to classify the samples from forget set $\mathcal{D}_f$ while maintaining the accuracy for classifying the samples from the retain set $\mathcal{D}_r$. This is ensured by using a small subset of samples $\mathcal{D}_{r\_\text{sub}}$ drawn from the retain dataset $\mathcal{D}_r$.

IV. Error-Maximizing Noise Based Unlearning

Our approach aims to learn a noise matrix for the unlearning class by maximizing the model loss. Such generated noise samples will damage/overwrite the previously learned network weights for the relevant class(es) during the model update and induce unlearning. Error maximizing noise will have high influence to enable parameters updates corresponding to the unlearning class.

A. Error-Maximizing Noise

We learn an error-maximizing noise $\mathcal{N}$ of the same size as that of the model input. The goal is to create a correlation between $\mathcal{N}$ and the unlearning class label, $f : \mathcal{N} \rightarrow \mathcal{Y}_f$, $\mathcal{N} \neq \mathcal{X}$. We freeze the weights of the pretrained model during this error maximizing process (see Fig. 1). Given a noise matrix $\mathcal{N}$, initialized randomly with a normal distribution $N(0, 1)$, we propose to optimize the error-maximizing noise by solving the following optimization problem:

$$\arg\min_{\mathcal{N}} \mathbb{E}_{(\theta)}\left[-\mathcal{L}(f, y) + \lambda \|w_{\text{noise}}\|\right] \tag{1}$$

where, $\mathcal{L}(f, y)$ is the classification loss corresponding to the class to unlearn, f denotes the trained model. The w_{noise} are the parameters of the noise $\mathcal{N}$ (can be interpreted as pixel values in terms of an image) and λ is used to manage the trade-off between the two terms. The optimization problem finds the L_p-norm bounded noise that maximizes the model's classification loss. In our method, we use a Cross-Entropy loss function $\mathcal{L}$ with L_2 normalization.

We maximize the error corresponding to the forget class(es) so that this noise is *opposite* to what D_f represents. Using this in the *impair* stage of the UNSIR algorithm erases information related to D_f. Overall, it enables efficient unlearning in deep networks. The second term $\lambda \|w_{\text{noise}}\|$ in (1) is proposed to regularize the overall loss by preventing the values in $\mathcal{N}$ from becoming too large. Without this regularization of $\mathcal{N}$, the model will start believing that images with higher values belong to the unlearn class. For multiple classes of data, we learn the noise matrix $\mathcal{N}$ for each class separately. Since the optimization is performed using the model loss with respect to the noise matrix, this can be done in an insignificant amount of time. The UNSIR algorithm will be executed only once for both single-class and multiclass unlearning.

B. UNSIR: Unlearning With Single Pass Impair and Repair

We combine the noise matrix along with the samples in $\mathcal{D}_{r\_\text{sub}}$, i.e., $\mathcal{D}_{r\_\text{sub}} \cup \mathcal{N}$, and train the model for one epoch (impair) to induce unlearning. After that we again train (repair) the model for one epoch, now on $\mathcal{D}_{r\_\text{sub}}$ only. The final model shows excellent performance in unlearning the targeted classes of data and retaining the accuracy on the remaining classes.

Impair: We train the model on a small subset of data from the original distribution which also contains generated noise. This step is called *impair* as it corrupts those weights in the network which are responsible for recognition of the data in forget class(es). We use a high learning rate and observe that almost always only a single epoch of *impair* is sufficient.

Repair: The *impair* step may sometimes disturb the weights that are responsible for predicting the retain classes. Thus, we *repair* those weights by training the model for a single epoch (on rare occasions, more epochs may be required) on the retain data $\mathcal{D}_{r\_\text{sub}}$. The final updated model has high RT, i.e., it takes substantial number of epochs for the network to relearn the forget samples. This is one of the important criteria for effective unlearning and the proposed method shows good robustness for the same. The overall framework of our unlearning algorithm is shown in Fig. 1.

V. Experiments and Results

We show the performance of our proposed method for unlearning single and multiples classes of data across a

variety of settings. We use different types of deep networks ResNet18 [46], AllCNN [47], MobileNetv2 [48] and ViTs [49] for evaluation and empirically demonstrate the applicability of our method across these different networks. The experiments are conducted for network trained from scratch as well as pretrained models fine-tuned on specific datasets. The unlearning method is analyzed over CIFAR-10 [50], CIFAR-100 [50] and VGGFace-100 (100 face IDs collected from the VGGFaces2 [51]). Results on these variety of models and datasets demonstrate the wide applicability of our method.

The experimental results are reported with a single step (one epoch) of *impair* and a single step of *repair*. Additional fine-tuning could be done, however, we focus on such a setting (single-shot) to demonstrate the efficacy of our method under a uniform setup. All the models learned from scratch have been trained for 40 epochs, and the pretrained models have been fine-tuned for five epochs. We observe that $\lambda = 0.1$ in (1) works quiet well across various tasks, and thus keep it fixed at 0.1 for all the experiments.

A. Evaluation Metrics

In the literature [3], [9], [10], [38] several metrics have been defined to measure the overall performance of an unlearning method. These metrics attempt to determine the amount of information remaining in the network about the unlearn/forget data. In our analysis, we use the following metrics.

Accuracy on Forget Set (A_{D_f}): Should be close to zero.

Accuracy on Retain Set (A_{D_r}): Should be close to the performance of original model.

RT: RT is a good proxy to measure the amount of information remaining in the model about the unlearning data. If a model regains the performance on the unlearn data very quickly with only few steps of retraining, it is highly likely that some information regarding the unlearn data is still present in the model. We measure the RT as the number of epochs it takes for the unlearned model to reach the source model's accuracy, with the model being trained on 500 random samples from the training set in each epoch.

Weight Distance: The distance between individual layers of the original model and the unlearned model gives additional insights about the amount of information remaining in the network about the forget data. A comparative analysis with the retrained model would validate the robustness of the unlearning method.

Prediction Distribution on Forget Class: We analyze the distribution of the predictions for different samples in the forget class(es) of data in the unlearned model. Presence of any specific observable patterns such as repeatedly predicting a single retain class may indicate risk of information exposure.

Additionally, a high similarity with the prediction distribution of the retrain model would indicate robustness in the unlearning method to information exposure of the forget class. A recent work [52] has reported the shortcomings of membership inference attacks on deep networks. Thus, we avoid using them to keep the analysis more consistent and reliable. It is to be noted that a comprehensive method of evaluating the exposure/leakage of private data in a deep model is a difficult task [38], and we are not aware of any method claiming to do so.

B. Models

In CIFAR-10, we trained *ResNet18* and *AllCNN* from scratch and used the proposed method to unlearn a single class and multiple classes (two classes, four classes, and seven classes) from the model. Without loss of generality, we use class 0 for single class unlearning, and a random manual selection of class subsets for multiclass unlearning. For example, in two-class unlearning we unlearn class 1 and 2, in four-class unlearning we unlearn classes 3–6, in seven-class unlearning we unlearn classes 3–9. In CIFAR-100, we use pretrained *ResNet18* and *MobileNetv2*. The unlearning is performed for one class (class 0), and 20 and 40 randomly sampled classes. In the latter part, we also demonstrate unlearning on VGGFace-100 using pretrained *ResNet18* and ViT. The unlearning is performed for 1-faceID, 20-faceID, 40-faceID, and 60-faceID, respectively.

C. Baseline Unlearning Methods

We primarily use the following baseline methods: 1) fine-tuning on the retain set, i.e., catastrophic forgetting (*FineTune*) and 2) gradient ascent on the forget class (*NegGrad*). The comparative results are shown in Table I. We also run *Fisher Forgetting* [9] and show the results in Table II. We present the results in two models for four-class forgetting as the Fisher method is computationally very expensive. We did not use methods such as removing the corresponding class from the final output as it does not remove any information from the model itself. Simply removing the final layer class might also lead to Streisand effect, i.e., the information we are trying to hide may become even more prominent.

D. Experimental Settings

The experiments are conducted on a NVIDIA Tesla-V100 (32 GB) GPU. The settings for individual datasets are given below:

CIFAR-10: The error-maximizing noise is learned for a single batch size and 20 copies of this noise are used for the noise dataset. A batch size of 256 is used for all the datasets. The retain set (D_r) is created by collecting 1000 samples of each retain class. The learning rate of 0.02 is used for impair step, where one epoch (one shot of damage) is trained done using the mix of retain sub-samples and noise. The learning rate in repair step is 0.01, where one epoch (one shot of healing) is trained on the retain sub-samples.

CIFAR-100: Same as in CIFAR-10, the error-maximizing noise is learned for a single batch size and 20 copies of this noise are used for the noise dataset. The retain set consists of 50 samples collected from each retain class. For pre-trained ResNet18, the learning rate in the impair step is set to 0.01 for the last layer and 0.0001 for the remaining layers. Likewise, in the repair step, the learning rate is set to 0.005 for the last layer and 0.0001 for rest of the layers. In the AllCNN model, the learning rate for impair and repair steps are 0.02 and 0.01, respectively.

TABLE I

UNLEARNING ON CIFAR-10. **ORIGINAL MODEL:** THE MODEL TRAINED ON COMPLETE DATASET D_c. **RETRAIN MODEL:** THE MODEL TRAINED ON RETAIN SET D_r. **FINE TUNE:** THE FINE TUNED MODEL ON D_r. **NEGGRAD:** THE NETWORK FINE TUNED ON D_f WITH NEGATIVE GRADIENTS (GRADIENT ASCENT). **OUR METHOD:** THE PROPOSED UNLEARNING METHOD. **RT:** RT IS THE *Number of Epochs* TAKEN BY MODEL TO REGAIN FULL ACCURACY ON FORGET SET WHEN TRAINED ON 500 RANDOM SAMPLES FROM D_c. A HIGHER VALUE OF **RT** DENOTES ROBUST ERASURE OF INFORMATION IN THE NETWORK WEIGHTS. THE ACCURACY A_{D_f} ON THE FORGET SET SHOULD BE CLOSE TO ZERO AND A_{D_r} SHOULD BE CLOSE TO ORIGINAL MODEL'S A_{D_r}. # $\mathcal{Y}_f$ DENOTES THE NUMBER OF UNLEARNING CLASSES

Model	# $\mathcal{Y}_f$	Metrics	Original Model	Retrain Model	FineTune [9]	NegGrad [9]	**Our Method**	Relearn Time (RT)			
								Retrain	FineTune [9]	NegGrad [9]	**Ours**
ResNet18	1	$A_{D_r}\uparrow$	77.86	78.32	78.11	66.67	**71.06 ±1.13**	77	8	54	90
		$A_{D_f}\downarrow$	81.01	0	24.55	7.44	**0 ±0**				
	2	$A_{D_r}\uparrow$	78.00	79.15	79.53	72.12	**73.61 ±0.51**	>100	10	7	>100
		$A_{D_f}\downarrow$	78.65	0	31.59	0.05	**0 ±0**				
	4	$A_{D_r}\uparrow$	81.42	85.88	85.49	54.84	**76.63 ±0.89**	>100	13	18	>100
		$A_{D_f}\downarrow$	73.45	0	41.45	0.02	**0 ±0**				
	7	$A_{D_r}\uparrow$	79.36	91.39	79.27	31.87	**82.86 ±1.42**	>100	0	>100	>100
		$A_{D_f}\downarrow$	77.58	0	77.71	0.19	**0 ±0**				
AllCNN	1	$A_{D_r}\uparrow$	82.64	85.90	85.01	42.39	**73.90 ±1.09**	>100	12	18	>100
		$A_{D_f}\downarrow$	91.02	0	31.07	13.47	**0 ±0**				
	2	$A_{D_r}\uparrow$	84.27	85.21	86.45	44.25	**80.76 ±1.33**	>100	6	7	74
		$A_{D_f}\downarrow$	79.74	0	35.45	2.20	**0 ±0**				
	4	$A_{D_r}\uparrow$	87.06	91.86	91.75	22.00	**80.21 ±0.73**	>100	4	24	>100
		$A_{D_f}\downarrow$	78.00	0	53.66	1.44	**0 ±0**				
	7	$A_{D_r}\uparrow$	83.30	94.47	83.29	22.96	**85.21 ±1.44**	>100	0	>100	>100
		$A_{D_f}\downarrow$	83.31	0	83.28	0	**0 ±0**				

TABLE II

COMPARISON OF OUR METHOD WITH A SINGLE CLASS FISHER FORGETTING [9] METHOD ON CIFAR-10. FISHER ACHIEVES FORGETTING BUT FAILS TO MAINTAIN THE ACCURACY ON THE RETAINED DATASET

	Initial Accuracy		Fisher Forgetting [9]		**Our Method**	
Model	$A_{D_f}\downarrow$	$A_{D_r}\uparrow$	$A_{D_f}\downarrow$	$A_{D_r}\uparrow$	$A_{D_f}\downarrow$	$A_{D_r}\uparrow$
ResNet18	81.01	77.86	0	10.85	**0**	**71.06**
AllCNN	91.02	82.64	0	7.61	**0**	**73.90**

VGGFace-100: A batch of the noise matrix is learned and copied 15 times to create the noise dataset. The retain set consists of 100 samples of each retain class. For ResNet18, the learning rate in impair and repair steps are 0.01 and 0.001, respectively. For ViT model, the learning rate for impair and repair steps are 0.0001 and 0.00002, respectively. We also run a Fisher Forgetting model as presented in [9] which is similar to a targeted noise addition based approach.

E. Results

Our results are compared with three baseline unlearning methods: Retrain Model, FineTune [9], and NegGrad [9]. We compare the single-class unlearning results with an existing Fisher forgetting method in Table II. Due to poor results of FineTune, NegGrad, and Fisher forgetting [9] in CIFAR-10, we compare our results only with the Retrain Model in the subsequent experiments.

1) Single Class Unlearning: Tables I and III show that our model is able to erase the information with respect to a particular class and unlearn in a single shot of impair and repair. Table I shows the retain and forget set accuracy after unlearning along with the average standard deviation after three runs. We obtain superior accuracy in retain set (D_r) and forget set (D_f) over the existing methods; such as in the case of ResNet18 and CIFAR-10, we preserve 71.06% of accuracy on D_r from an initial 77.86% while degrading the performance on D_f significantly (0% from an initial 81.01%). The RT is much higher for our method in comparison to the baseline methods (for example, >100 versus 12, 18 in the case of AllCNN, CIFAR-10). This shows the capability of our method to enforce robust unlearning. From Table II, we observe that our method is far superior to Fisher forgetting as well. Fisher Forgetting is able to preserve only 10.85% accuracy in ResNet on D_r on CIFAR-10.

TABLE III

UNLEARNING ON CIFAR-100. THE MODELS ARE PRETRAINED ON ImageNET AND FINE TUNED FOR CIFAR-100

Model	# $\mathcal{Y}_f$	Metrics	Original Model	Retrain Model	**Our Method**
ResNet18	1	$A_{D_r}\uparrow$	78.68	78.37	**75.36**
		$A_{D_f}\downarrow$	83.00	0	**0**
	20	$A_{D_r}\uparrow$	77.88	79.73	**75.38**
		$A_{D_f}\downarrow$	82.84	0	**0**
	40	$A_{D_r}\uparrow$	78.31	82.65	**78.85**
		$A_{D_f}\downarrow$	79.78	0	**0**
	60	$A_{D_r}\uparrow$	76.96	83.62	**75.51**
		$A_{D_f}\downarrow$	80.31	0	**0.47**
MobileNetv2	1	$A_{D_r}\uparrow$	77.43	78	**75.76**
		$A_{D_f}\downarrow$	90	0	**0**
	20	$A_{D_r}\uparrow$	76.47	77	**76.27**
		$A_{D_f}\downarrow$	81.70	0	**0**
	40	$A_{D_r}\uparrow$	76.93	80.24	**77.66**
		$A_{D_f}\downarrow$	78.56	0	**0.02**
	60	$A_{D_r}\uparrow$	76.17	79.37	**68.57**
		$A_{D_f}\downarrow$	78.56	0	**1.22**

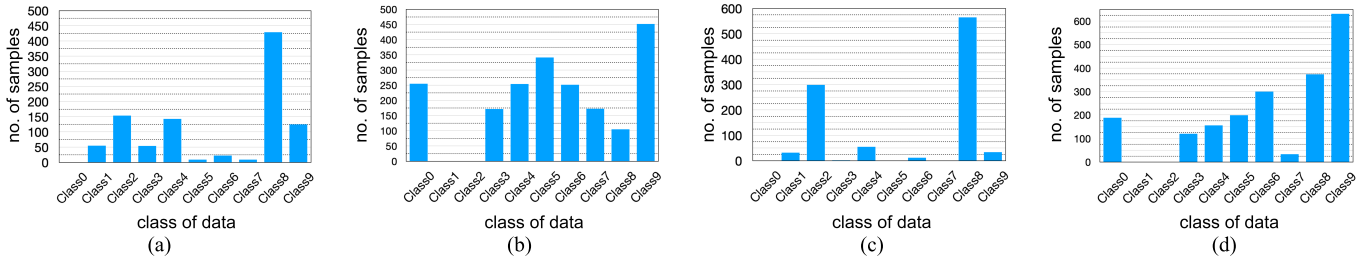

Fig. 2. Prediction distribution of the unlearned model on forget class of data. Our method gives randomized response to the input query of the forget class of data. (a) 1-C unlearning (AllCNN). (b) 2-C unlearning (AllCNN). (c) 1-C unlearning (ResNet18). (d) 2-C unlearning (ResNet18).

TABLE IV

UNLEARNING ON VGGFACE-100. THE MODELS ARE PRETRAINED ON IMAGENET AND FINE TUNED FOR VGGFACE-100

Model	# $\mathcal{Y}_f$	Metrics	Original Model	Retrain Model	**Our Method**
ResNet18	1	$A_{D_r} \uparrow$	80.63	80.42	**72.79**
		$A_{D_f} \downarrow$	94.00	0	**3.00**
	20	$A_{D_r} \uparrow$	81.15	69.96	**73.26**
		$A_{D_f} \downarrow$	78.45	0	**0.15**
	40	$A_{D_r} \uparrow$	81.31	82.74	**78.66**
		$A_{D_f} \downarrow$	79.18	0	**6.71**
	60	$A_{D_r} \uparrow$	81.30	82.66	**79.16**
		$A_{D_f} \downarrow$	80.03	0	**8.62**
ViT	1	$A_{D_r} \uparrow$	91.53	92.45	**82.90**
		$A_{D_f} \downarrow$	74.22	0	**4.81**
	20	$A_{D_r} \uparrow$	91.52	93.70	**85.21**
		$A_{D_f} \downarrow$	91.30	0	**26.00**
	40	$A_{D_r} \uparrow$	92.10	94.13	**85.33**
		$A_{D_f} \downarrow$	90.55	0	**25.10**
	60	$A_{D_r} \uparrow$	90.97	93.35	**87.82**
		$A_{D_f} \downarrow$	91.82	0	**8.48**

2) Multiple Class Unlearning: Our method shows the excellent unlearning result for multiclass unlearning. We observe that as the number of classes to unlearn increases, the repair step becomes more effective and leads to performance closer to the original model on D_r. The experiments are done with pretrained ResNet18 on CIFAR-100. After unlearning 20 classes we retain 75.38% accuracy compared to an initial 77.88% on retain set. The FineTune and gradient ascent (NegGrad) methods either lose performance on D_r or their performance on D_f is much higher than expected. For example, in the case of four-class unlearning on CIFAR-10, FineTune retains decent accuracy on D_r but it fails to unlearn D_f properly. It preserves 53.66% accuracy on the forget classes versus 0% preserved by our method. The NegGrad appears to unlearn the forget classes properly but its performance on D_r takes a hit. It obtains 22% retain set accuracy versus 80.21% accuracy obtained by our method. In addition, our method significantly outperforms both FineTune and NegGrad in RT. This suggests that much of the information about D_f is still present in the unlearned model which is not desirable. For example, in case of two-class unlearning on CIFAR-10 + ResNet-18, NegGrad achieves a decent 72.12% on D_r and 0.05% on D_f. But the model relearns in seven epochs compared to our method's RT of 100 epochs. Thus, our method shows excellent overall unlearning results as reported in Tables I, III, IV for multiclass unlearning.

3) Unlearning in Face Recognition: Facial images are difficult to differentiate from each other for a model and are one of the most challenging unlearning tasks. The results on VGGFace-100 is obtained using ResNet and pretrained ViT and reported in Table IV. We report the unlearning performance on D_r and D_f after forgetting 1 class, 20 classes, 40 classes, and 60 classes. As the ViT model is obtained with a few epochs (five epochs) of fine-tuning, the RT is expected to be low as well. Therefore, we do not present the analysis corresponding to the RT. Our method achieves good retention as well as forgetting accuracy. Such as for one class forgetting on ResNet18, our method preserves 72.29% accuracy on D_r compared to an initial 80.63% and degrades the performance on D_f to 3%. In case of 60 classes forgetting on pretrained ViT, our method preserves 87.82% accuracy (initial accuracy: 90.97%) on D_r and 8.48% (initial accuracy: 91.82%) on the D_f. This showcases the wide applicability of our method.

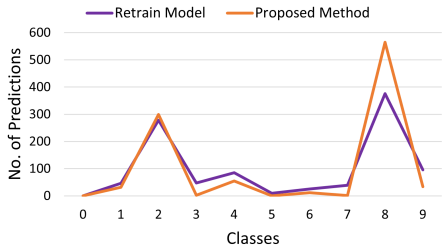

Fig. 3. Prediction distribution (AllCNN) of the retrained model (left) and proposed method (right) for the forget class. We can see the distributions are similar.

F. Prediction Distribution for the Forget Class of Data

We plot the graph of the prediction class outcomes of the unlearned model for the forget class of data. For example, Fig. 2(a) depicts the prediction outcomes of an unlearned ResNet18 model (*forget class = class0*) for the samples from *class0*. The prediction outcomes for two-classes unlearned model (*forget class = class1, class2*) is also shown in Fig. 2(b). Here we can check whether our unlearned model predicts a specific class(es) for all the forget set of data (Streisand effect [9]) because this could lead to a potential vulnerability to adversarial attacks. We observe in Fig. 2 that all the predictions for the forget class of data are randomly distributed across the remaining retain classes. Our unlearned model is unable to confidently correlate the forget data with any specific retain class. This shows that our method has actually erased the information related to the unlearn class of data.

We also compare the predictions of the retrained model (gold model) and the proposed method in Fig. 3. It can be

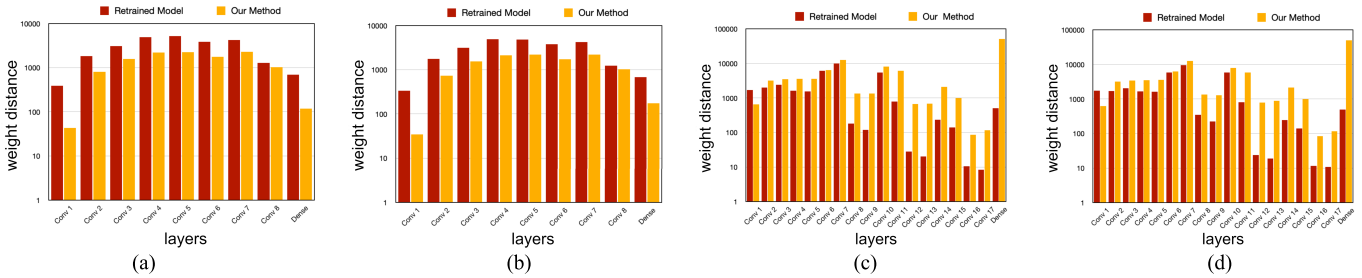

Fig. 4. Layer-wise weight distance between the unlearned models (retrain model, our model) and the original model. The values are presented on a log scale. Our method obtains comparable or higher weight distances in comparison to the retrain model. (a) 1-C unlearning (AllCNN). (b) 2-C unlearning (AllCNN). (c) 1-C unlearning (ResNet18). (d) 2-C unlearning (ResNet18).

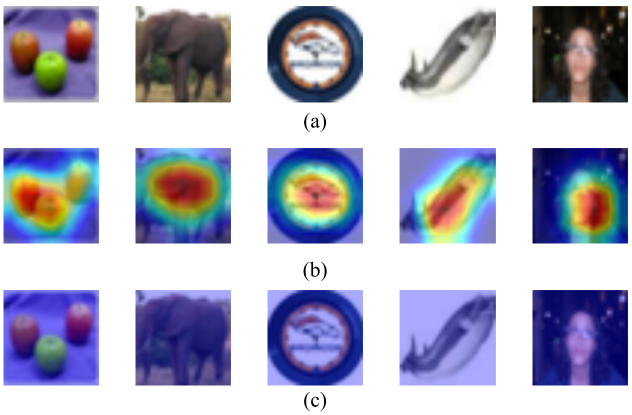

Fig. 5. GradCAM visualization of ResNet18 on CIFAR-100. The first column depicts visualization in one-class unlearning and the remaining columns depict the visualization in four-classes unlearning. (a) Input. (b) Original model. (c) Unlearned model.

observed that the output distribution in both models is very similar. This further shows the robustness of our method.

VI. Analysis

A. Layer-Wise Distance Between the Network Weights

The layer-wise distance between the original and unlearned models help in understanding the effect of unlearning at each layer. The weight difference should be comparable to the retrain model as a lower distance indicates ineffective unlearning and a much higher distance may point to Streisand effect and possible information leaks. We compare the weight distance in the 1) retrained model, and 2) proposed method for AllCNN and ResNet18 in Fig. 4. We notice that the weight differences of the proposed method with respect to the original model show a similar trend to that of the retrain model.

B. Visualizing the Unlearning in Models

We use GradCAM [53] to visualize the area of focus in the model (ResNet18) for images in the unlearn class. Fig. 5 depicts where the model focuses before and after applying our method for unlearning one-class and 20-classes, respectively. As expected, after applying our method, the model is unable to focus on the relevant areas, indicating that the network weights no longer contain information related to those unlearn classes.

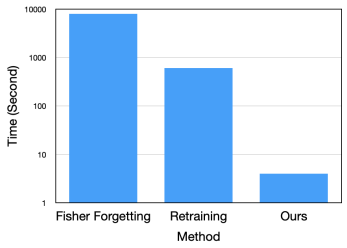

Fig. 6. Figure shows the training time comparison between Retraining, Fisher Forgetting, and UNSIR (our method).

C. Efficiency

Our method is fast and highly efficient in comparison to retraining and the existing unlearning approaches [9], [10]. The Fisher Forgetting [9] and NTK-based forgetting [10] approaches require Hessian approximation which is computationally very expensive. These methods give some bounds on the amount of information remaining but they are quite inefficient for practical use. They take even more time than retraining itself. Whereas retraining took us around 10 min (617 s), it took us more than 2 h to run Fisher forgetting [9] for one-class unlearning in ResNet18 + CIFAR-10. The Fisher forgetting for one-class unlearning in AllCNN + CIFAR-10 takes around 1 h. For CIFAR-100, the estimated time surpassed 25 h. The NTK-based forgetting [10] uses Fisher noise along with NTK-based model approximations and thus is even more time-consuming. Our method only requires 1.1 s for 40 steps of noise optimization on ResNet18 + CIFAR-10, 1.70 s for one epoch of impair, and 1.13 s for an epoch of repair. The total computational time for unlearning is less than 4 s. This is 154× faster than the retraining approach, 1875× faster than the Fisher approach. We achieve fast unlearning without compromising the effectiveness of the method. Moreover, *our method is scalable* to large problems and big models. The cost of noise matrix estimation depends on the cost of a forward pass in the model. Usually, in multiclass unlearning, the cost of noise matrix estimation is linearly dependent on the number of forget classes. In the case of UNSIR, the algorithm is executed only once for both single-class or multiclass unlearning. Thus, our method offers the most efficient multiple-class unlearning among them. Fig. 6 shows the time complexity comparison for retraining, Fisher Forgetting, and UNSIR. Our method requires 1250× less time than retraining and 125× less time than Fisher forgetting.

TABLE V
OBSERVING THE EFFECT OF DIFFERENT COMBINATION OF IMPAIR-REPAIR STEPS. THE EXPERIMENTS ARE DONE ON RESNET18 + VGGFACE-100

Setting	Intermediate	Accuracy on Forget Set	Accuracy On Retain Set
1×(Impair-Repair)	Before Impair	80.63	94
	After Impair	2	53
	After Repair	3	72.79
2×(Impair)-1×(Repair)	Before Impair	80.63	94
	After 2 Impairs	0	49.05
	After Repair	3	72.5
2×(Impair-Repair)	Before Impair	80.63	94
	After Cycle 1	3	72.79
	After Cycle 2	0	70.86

D. Comparing Different Impair-Repair Configurations

We conduct experiments to provide a comparison between different impair-repair configurations on ResNet18 + VGGFace-100 in Table V. A single impair-repair cycle does not yield the expected 0% accuracy on forget set. Since most of the damage is done in the impair step, we observe the effect of executing two impair steps before the repair step. After impair, the performance on the forget set reaches the desired 0% but the model regains 3% accuracy after the repair step. We then execute two cycles of impair-repair. This means one impair, one repair, one impair, and one repair step. This yields the expected 0% on the forget set with minimal loss on performance on the retain set (72.79, 72.5 versus 70.86%). Furthermore, additional ablation analysis is presented in the Supplementary Material.

E. Limitations

Our method achieves unlearning in already trained deep learning models. The existing approaches [3], [9], [10], [12] either require training the models in a specific manner or make impractical assumptions such as linear models or treating deep model training as a convex optimization problem and are thus incompatible with our target settings of unlearning from an already trained model. The unlearning approach in [4] provides an exact unlearning guarantee but consumes a lot of memory and requires implementation during the training process. Our method can be used to perform unlearning as an afterthought, i.e., delete data from previously deployed deep learning models. Similarly, unlike the linear/convex case, where strong bounds on the amount of remaining information can be formulated, forgetting on DNNs often does not come with any provable bound. This is still an open problem. The kind of information bounds given in the above works are not compatible with our framework. To cope with this limitation, we conduct extensive experimental analysis to check the unlearning performance through a variety of widely accepted metrics. We use performance on retain and forget set, layer-wise weight difference, prediction distribution comparison for forget set and RT to evaluate the unlearning and showcase that our method is effective with no empirical signs of information leakage. However, a more formal guarantee of unlearning might be desired in highly privacy-sensitive applications.

Unlearning a random cohort of data is beyond the scope of this work. Although, in theory, an error-maximizing noise matrix can be generated corresponding to the random samples. But this would hurt the zero-glance assumption and thus, unlearning random samples, or only a subset of a class is out of the scope of this article. Furthermore, the analysis of adaptive adversaries, that have exact knowledge about the proposed algorithm, is out of scope as well. We also point out the trade-off between speed and accuracy in our unlearning method. For example, the efficiency gain in the proposed method is 154× and 1875× more than the retrain and Fisher method, respectively. However, this efficiency gain comes at the cost of decreased accuracy in comparison to the retrain method.

VII. CONCLUSION

In this article, we presented a stringent *zero-glance* setting for unlearning and explore an efficient solution for it. We also develop a scalable, multiple-class unlearning method. The unlearning method consists of learning an error-maximizing noise matrix followed by single pass impair and repair to update the network weights. Different from existing works, our method is highly efficient in unlearning multiple classes of data and we empirically demonstrate its effectiveness in a variety of deep networks such as CNN and ViT. The method is applicable to deep networks trained with any kind of optimization. Excellent unlearning results on a large-scale face recognition dataset are also shown which is the first such attempt. Our work opens up a new direction for efficient multiclass unlearning on large-scale problems. A possible future direction could be to perform unlearning without using any kind of training samples.

ACKNOWLEDGMENT

This research/project is supported by the National Research Foundation, Singapore under its Strategic Capability Research Centres Funding Initiative. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

REFERENCES

[1] P. Voigt and A. Von Dem Bussche, "The EU general data protection regulation (GDPR)," in *A Practical Guide*, 1st ed. Cham, Switzerland: Springer, 2017.
[2] E. Goldman, "An introduction to the California consumer privacy act (CCPA)," in *Proc. Santa Clara Univ. Legal Stud. Res. Paper*, 2020, pp. 1–7.
[3] A. Golatkar, A. Achille, A. Ravichandran, M. Polito, and S. Soatto, "Mixed-privacy forgetting in deep networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 792–801.
[4] L. Bourtoule et al., "Machine unlearning," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2021, pp. 141–159.
[5] D. M. Sommer, L. Song, S. Wagh, and P. Mittal, "Athena: Probabilistic verification of machine unlearning," *Proc. Privacy Enhancing Technol.*, vol. 2022, no. 3, pp. 268–290, Jul. 2022.
[6] S. Garg, S. Goldwasser, and P. N. Vasudevan, "Formalizing data deletion in the context of the right to be forgotten," in *Proc. Adv. Cryptol.—EUROCRYPT*, Zagreb, Croatia, May 2020, pp. 373–402.
[7] J. Brophy and D. Lowd, "Machine unlearning for random forests," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 1092–1104.
[8] Q. P. Nguyen, B. K. H. Low, and P. Jaillet, "Variational Bayesian unlearning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 16025–16036.
[9] A. Golatkar, A. Achille, and S. Soatto, "Eternal sunshine of the spotless net: Selective forgetting in deep networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 9304–9312.

[10] A. Golatkar, A. Achille, and S. Soatto, "Forgetting outside the box: Scrubbing deep networks of information accessible from input-output observations," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 383–398.

[11] A. Ginart, M. Y. Guan, G. Valiant, and J. Zou, "Making ai forget you: Data deletion in machine learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 3513–3526.

[12] C. Guo, T. Goldstein, A. Hannun, and L. Van Der Maaten, "Certified data removal from machine learning models," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 3832–3842.

[13] T. Baumhauer, P. Schöttle, and M. Zeppelzauer, "Machine unlearning: Linear filtration for logit-based classifiers," *Mach. Learn.*, vol. 111, no. 9, pp. 3203–3226, Sep. 2022.

[14] Z. Izzo, M. A. Smart, K. Chaudhuri, and J. Zou, "Approximate data deletion from machine learning models," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2021, pp. 2008–2016.

[15] S. Neel, A. Roth, and S. Sharifi-Malvajerdi, "Descent-to-delete: Gradient-based methods for machine unlearning," in *Proc. 32nd Int. Conf. Algorithmic Learn. Theory*, 2021, pp. 931–962.

[16] Y. Wu, E. Dobriban, and S. Davidson, "DeltaGrad: Rapid retraining of machine learning models," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 10355–10366.

[17] V. S. Chundawat, A. K. Tarun, M. Mandal, and M. Kankanhalli, "Can bad teaching induce forgetting? Unlearning in deep networks using an incompetent teacher," in *Proc. AAAI Conf. Artif. Intell.*, 2023, pp. 1–12.

[18] A. K. Tarun, V. S. Chundawat, M. Mandal, and M. Kankanhalli, "Deep regression unlearning," 2022, *arXiv:2210.08196*.

[19] V. S. Chundawat, A. K. Tarun, M. Mandal, and M. Kankanhalli, "Zero-shot machine unlearning," *IEEE Trans. Inf. Forensics Inf. Security*, early access, Apr. 7, 2023, doi: 10.1109/TIFS.2023.3265506.

[20] A. Mahadevan and M. Mathioudakis, "Certifiable machine unlearning for linear models," 2021, *arXiv:2106.15093*.

[21] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun, "The loss surfaces of multilayer networks," in *Proc. 18th Int. Conf. Artif. Intell. Statist.*, 2015, pp. 192–204.

[22] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1885–1894.

[23] P. W. Koh, K.-S. Ang, H. Teo, and P. S. Liang, "On the accuracy of influence functions for measuring group effects," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 5254–5264.

[24] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *Stat*, vol. 1050, p. 20, Dec. 2015.

[25] H. Huang, X. Ma, S. M. Erfani, J. Bailey, and Y. Wang, "Unlearnable examples: Making personal data unexploitable," in *Proc. ICLR*, 2021, pp. 1–17.

[26] Y. Cao and J. Yang, "Towards making systems forget with machine unlearning," in *Proc. IEEE Symp. Secur. Privacy*, May 2015, pp. 463–480.

[27] B. Mirzasoleiman, A. Karbasi, and A. Krause, "Deletion-robust submodular maximization: Data summarization with 'the right to be forgotten,"' in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2449–2458.

[28] M. Abadi et al., "Deep learning with differential privacy," in *Proc. 2016 ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 308–318.

[29] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, nos. 3–4, pp. 211–407, Aug. 2014.

[30] V. Gupta, C. Jung, S. Neel, A. Roth, S. Sharifi-Malvajerdi, and C. Waites, "Adaptive machine unlearning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 16319–16330.

[31] A. Prabhu, P. H. Torr, and P. K. Dokania, "GDumb: A simple approach that questions our progress in continual learning," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2020, pp. 524–540.

[32] A. Achille, G. Paolini, and S. Soatto, "Where is the information in a deep neural network?" 2019, *arXiv:1905.12213*.

[33] M. Chen, Z. Zhang, T. Wang, M. Backes, M. Humbert, and Y. Zhang, "When machine unlearning jeopardizes privacy," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2021, pp. 896–911.

[34] N. G. Marchant, B. I. Rubinstein, and S. Alfeld, "Hard to forget: Poisoning attacks on certified machine unlearning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 36, no. 7, 2022, pp. 7691–7700.

[35] A. Thudi, H. Jia, I. Shumailov, and N. Papernot, "On the necessity of auditable algorithmic definitions for machine unlearning," in *Proc. 31st USENIX Secur. Symp. (USENIX Secur.)*, 2022, pp. 4007–4022.

[36] C. Wu, S. Zhu, and P. Mitra, "Federated unlearning with knowledge distillation," 2022, *arXiv:2201.09441*.

[37] C. Chen, F. Sun, M. Zhang, and B. Ding, "Recommendation unlearning," in *Proc. ACM Web Conf.*, Apr. 2022, pp. 2768–2777.

[38] L. Graves, V. Nagisetty, and V. Ganesh, "Amnesiac machine learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 13, 2021, pp. 11516–11524.

[39] A. Sekhari, J. Acharya, G. Kamath, and A. T. Suresh, "Remember what you want to forget: Algorithms for machine unlearning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021.

[40] T. Shibata, G. Irie, D. Ikami, and Y. Mitsuzumi, "Learning with selective forgetting," in *Proc. 30th Int. Joint Conf. Artif. Intell.*, vol. 2, no. 4, 2021, p. 6.

[41] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proc. 53rd Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2015, pp. 1310–1321.

[42] N. Phan, Y. Wang, X. Wu, and D. Dou, "Differential privacy preservation for deep auto-encoders: An application of human behavior prediction," in *Proc. 30th AAAI Conf. Artif. Intell.*, 2016, pp. 1–8.

[43] S. Shan, E. Wenger, J. Zhang, H. Li, H. Zheng, and B. Y. Zhao, "Fawkes: Protecting privacy against unauthorized deep learning models," in *Proc. 29th USENIX Secur. Symp. (USENIX Secur.)*, 2020, pp. 1589–1604.

[44] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1765–1773.

[45] Z. Shen, S. Fan, Y. Wong, T.-T. Ng, and M. Kankanhalli, "Human-imperceptible privacy protection against machines," in *Proc. 27th ACM Int. Conf. Multimedia*, Oct. 2019, pp. 1119–1128.

[46] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[47] J. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," in *Proc. ICLR (Workshop Track)*, 2015, pp. 1–14.

[48] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.

[49] A. Dosovitskiy et al., "An image is worth 16×16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 1–22.

[50] A. Krizhevsky et al., "Learning multiple layers of features from tiny images," CIFAR, Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009.

[51] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, "VGGFace2: A dataset for recognising faces across pose and age," in *Proc. 13th IEEE Int. Conf. Autom. Face Gesture Recognit. (FG)*, May 2018, pp. 67–74.

[52] S. Rezaei and X. Liu, "On the difficulty of membership inference attacks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 7892–7900.

[53] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 618–626.