Learn to Unlearn for Deep Neural Networks: Minimizing Unlearning Interference with Gradient Projection

Tuan Hoang   Santu Rana   Sunil Gupta   Svetha Venkatesh
A2I2, Deakin University
{tuan.h;santu.rana;sunil.gupta;svetha.venkatesh}@deakin.edu.au

Abstract

Recent data-privacy laws have sparked interest in machine unlearning, which involves removing the effect of specific training samples from a learnt model as if they were never present in the original training dataset. The challenge of machine unlearning is to discard information about the "forget" data in the learnt model without altering the knowledge about the remaining dataset and to do so more efficiently than the naive retraining approach. To achieve this, we adopt a projected-gradient based learning method, named as Projected-Gradient Unlearning (PGU), in which the model takes steps in the orthogonal direction to the gradient subspaces deemed unimportant for the retaining dataset, so as to its knowledge is preserved. By utilizing Stochastic Gradient Descent (SGD) to update the model weights, our method can efficiently scale to any model and dataset size. We provide empirically evidence to demonstrate that our unlearning method can produce models that behave similar to models retrained from scratch across various metrics even when the training dataset is no longer accessible. Our code is available at https://github.com/hnanhtuan/projected_gradient_unlearning.

1. Introduction

Deep learning has widely adopted across various fields, such as computer vision, natural language processing, and image/music/video generation. One of the reasons deep models excel is their ability to leverage vast quantities of data for training. However, machine learning models may unintentionally memorize their training data to a certain level, and recent work has shown that it is possible to derive meaningful information about individual training examples using only the parameters of a trained model [7, 11, 31, 40, 45]. When the training data potentially contains privacy-sensitive user information, this creates significant challenges in regulating access to each user's data or enforcing personal data ownership, which the General Data Protection Regulation (GDPR) in the European Union [28] and the California Consumer Privacy Act (CCPA) [34] aim to address. Therefore, it becomes imperative to develop learning techniques that limit such memorization (such as Differential Privacy learning) or remove such model memorization when necessary (through Machine Unlearning), which is the main focus of this paper. Such a problem of machine unlearning can extends to the other applications such as de-poisoning, where we want to remove the effect of a subset of data previously used for training and later identified as malicious (e.g., anomalies) [5] or biased [29] (denoted as "poisoned samples"). After the unlearning process, the ML model ideally performs well as if the model has not been trained with the malicious/biased data.

When users invoke their *"right to be forgotten" (RTBF)*, it's crucial to ensure that the user data is "unlearnt" from the trained model. This means that any information derived from the requested-to-delete (forgetting) data should be removed from the model's knowledge. The first plausible solution is to retrain the model from scratch without including the deleted data. However, this approach may be less practical due to its *high computation, time*, and *space costs*. Furthermore, this solution may sometimes require re-collection of training data to retrain the model as the training data may not be stored indefinitely due to privacy regulations [28]. For the same reason, it would be necessary for the unlearning method to work without requiring the training data.

Numerous methods [1, 3, 12, 14, 15, 18, 20, 22, 30, 32, 33, 44] have been proposed to facilitate unlearning. Many of these works [1, 15, 18, 30, 44] rely on the influence function [23], that helps to estimate the influence of training data on the trained models, to find the update that reverse the effect of forgetting samples on this model. However, these works tend to be computationally intensive due to the Hessian estimation. As a result, it is difficult to achieve a significant runtime improvement for large models such as CNN over retraining.

In order to address the challenges of forgetting training knowledge with more computational efficiency, we first introduce a novel unlearning loss for classification which aims to reverse the original training process of the forgetting data.

Then, inspired by recent works [27, 38], we apply orthogonal gradient steps with respect to the core gradient subspace of the model weights for the retaining dataset. Specifically, we partition the entire gradient space of weights into two orthogonal subspaces: Core Gradient Space (CGS) and Residual Gradient Space (RGS) [37], where CGS contains information that needs to be preserved. This approach enables us to remove information related to the forgetting data from the trained model while inducing minimum interference with the retaining dataset, thereby avoiding catastrophic forgetting. We also propose a technique that can efficiently compute the CGS of the model weights for the retaining dataset from only the weight gradient space of the full training data, which is particularly useful when the full training dataset is no longer accessible.

Our contributions can be summarized as follows: **(i)** We propose an unlearning method for classification task with a novel unlearning loss function. The method utilizes gradient projection to remove information from a trained model with minimum interfering to important information of the retaining data, thereby preventing prevent catastrophic forgetting. **(ii)** The proposed method only requires the forgetting data during the unlearning process and is applicable even when the training data is no longer accessible. **(iii)** In addition, our work can be applied to depoisoning application to eliminate harmful effects of poisoned training samples. **(iv)** As our method employs gradient descent updates to unlearn the model, our method can be scaled effortlessly to any model and dataset. **(v)** Our experiment results on large scale models and datasets demonstrate that our unlearning method can produce models that behave similarly to models retrained from scratch across various metrics.

2. Related works

2.1. Machine unlearning

Earlier works on Machine Unlearning has been studied on the exact unlearning such as SVM [21, 35], Naive Bayes classifiers, and k-means. However, these approaches are not suitable for CNNs, which are trained using stochastic gradient descent.

Yinzhi *et al.* [5] shows an efficient forgetting algorithm in the restricted setting of statistical query learning, where the learning algorithm cannot access individual samples. Bourtoule *et al.* [4] introduce the ”sharded, isolated, sliced, and aggregated” (SISA) framework as a low-cost solution for knowledge removal, which involves the following three steps: (1) partition the complete training sample set into multiple disjoint shards, (2) train models independently on each of these shards, and (3) retrain the affected model upon receiving a request to unlearn a training point. However, this approach may incur a large storage overhead and its efficiency quickly deteriorates when multiple data points need to be removed. Gou *et al.* [18] propose a certified-removal mechanism, a very strong theoretical guarantee that an unlearned model is indistinguishable from a retrained model that never encountered the data in the first place. The method utilizes the influence function [23] for L2-regularized linear models that are trained using a differentiable convex loss function, such as logistic regressors. However, the method does not extend to DNN due to its strong convex assumption. Golatkar *el at.* [1] propose a selective forgetting procedure for DNNs trained with SGD, using an information theoretic formulation and exploiting the stability of SGD. They propose a forgetting mechanism which involves a shift in weight space, and addition of noise to the weights to destroy information. [15] proposes a scrubing method by adopting Neural Tangent Kernel (NTK), which posits that large networks during training evolve in the same way as their linear approximations [26]. This allows the model information can be scrubbed in one step (i.e., “one-shot forgetting”). However, this method faces the computational bottleneck as the size of NTK matrix grows exponentially with the number of training samples and classes. Based on the forgetting theory provided by Sekhari *et. al.* [39], Mehta et. al. [30] propose a measure for computing conditional independence called L-CODEC which identifies the Markov Blanket of parameters to be updated so that the method can be applied to large models. However, the method can still be computationally-intensive for a very deep and wide network. Baumhauer *et al.* [3] introduce a forgetting method for logit-based classification models by applying a linear transformation to the output logits. However, this method only applies *filtration* to the final linear layer, and other layer weights may still retain information. Therefore, this method may not effectively address data privacy concerns. Additionally, this approach has limited applicability, as it can only be used for class-wide data deletion. In the work by Kurmanji *et al.* [25], they introduce a novel approach to unlearning through the optimization of a *min-max* problem. Within this framework, the *max* steps are designed to guide the student model to to distance its outputs from the teacher outputs on forgotten examples, effectively erasing forgotten information. While the *min* steps align the student model outputs with the teacher model outputs on retained examples for restoring the model's performance on retained data, should it have been adversely affected by the *max* steps. Thudi *et. al.* [43] design of a new training objective penalty that limits the overall change in weights during SGD and as a result facilitates approximate unlearning. Zhang *et. al.* [48] initially establish a connection between randomized smoothing techniques for achieving certified robustness in classification tasks and randomized smoothing methods for certified machine unlearning with gradient quantization. Then based on that connection, they propose the concept of Prompt Certified Machine Unlearning (PCMU), which is built upon a foundation of randomized data smoothing and gradient quantization.

2.2. Differential privacy

Differential Privacy (DP) [2,9,10,16,17] offers a formal solution to address data privacy concerns and safeguard data ownership. Compared to unlearning, DP is more stringent as it restricts memorization and seeks to learn model parameters in a way that prevents retrieval of any information related to any training samples, while still achieving reasonable performance. On the other hand, unlearning merely aims to remove model information associated with a subset of training data after standard training, without expecting the model to perform well on those deleted samples. Due to its more rigorous requirements, differential privacy for deep networks can be challenging to achieve and often results in significant accuracy losses. Therefore, when the requirement about data privacy is not excessively strict, machine unlearning may be a more suitable option.

2.3. Membership Inference Attack

Membership Inference Attack (MIA) [6,8,36,40] tries to determine if a particular data was used for training a model. This attack can serve as an effective means of evaluating the forgetting capacity of a model, especially when there are no or weak theoretical guarantees to quantify the remaining knowledge of forgetting data in model parameters. If the attacker's predictions are comparable for both unlearned and retrained models, then it implies that the unlearned model has lost information that is specific to the forgetting data. In contrast, if the attacker displays a greater degree of confidence in predicting a forgetting sample from an unlearned model as opposed to a retrained one, it could indicate ineffective unlearning. Conversely, a lower confidence than random chances in predicting from an unlearned model could lead to the Streisand Effect.

3. Proposed method

3.1. Problem statement

Let $\mathcal{D} = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^N$ be a fixed training dataset and $f_{\mathbf{w}}(\boldsymbol{x})$ be a parametric function (model), for instance a CNN, with parameters $\mathbf{w}$ (weights) trained on $\mathcal{D}$. Let $\mathcal{D}_f \subset \mathcal{D}$ be a subset of the training data, whose information we want to remove from the model $f_{\mathbf{w}}(x)$ (i.e., *forgetting dataset*), and let $\mathcal{D}_r$ be the complement of $\mathcal{D}_f$ (i.e., $\mathcal{D}_f \cup \mathcal{D}_r = \mathcal{D}$ and $\mathcal{D}_f \cap \mathcal{D}_r = \emptyset$), whose information we want to retain (i.e., *retaining dataset*).

3.2. Unlearning

Our approach leverages the property that stochastic gradient descent (SGD) updates lie in the span of input data points [47]. Inspired by the works of Schulman *et al.* (2015) [38] and Lin *et al.* [27], we have developed a method to selectively forget parts of a dataset by applying gradient updates orthogonal to the Core Gradient Space (CGS) [37] of model weights computed using the retaining data. This approach allows the model to update its weights in a way that discards information about the forgetting data while preserving the knowledge learnt from the retaining data.

3.2.1 Loss function:

We first propose the loss function that reverses the learning process as follows:

$$\mathcal{L} = \sum_{i \in \mathcal{D}_f} \sum_{c=1}^{C} \left(-y_{i,c} \log(1 - p_{i,c} + \epsilon) - \lambda p_{i,c} \log(p_{i,c}) \right), \tag{1}$$

where $p_{i,c} = \frac{\exp(z_{i,c})}{\sum_{j=1}^{C} \exp(z_{i,j})}$; $z_{i,c}$ is the c-th element of $\boldsymbol{z}_i = f_{\mathbf{w}}(\boldsymbol{x}_i)$, and similarly $y_{i,c}$ is the c-th element of $\boldsymbol{y}_i$.

The first-term is "*reverse*" cross-entropy loss which tries to *minimize* the predicted probability of the true-label class. ϵ is a small constant value to avoid exponentially large gradients at the beginning of training when the scores can be high (i.e., $p_{i,c} \approx 1$). The second-term attempts to *maximize* ($\lambda > 0$) the entropy of the model outputs, thereby making equal confidence scores for all classes. This term is helpful in removing information from the forgetting data that could be useful in multiple classes (e.g., bird and airplane might share similar background information). In other words, the model also become less focused on features that are correlated with the forgetting labels. Moreover, the second term also prevents the confidence scores of the forgetting data from going arbitrarily close to 0, which could otherwise result in abnormal confidence scores for the forgetting data.

For the de-poisoning application, we not only want the model to unlearn the features (which could be noise) that are correlated with the ”poisoned” labels, but we also expect the model to correctly re-classify these poisoned samples. Therefore, we use $\lambda < 0$ to minimize the entropy of the model outputs. This means that we want the model to assign a high confidence score to a class other than the poisoned labels.

3.2.2 Core Gradient Space (CGS) construction:

When the model training on full dataset is finished, we then compute the basis vectors and eigen-values of the gradient space for full dataset. Specifically, for each convolutional or linear layer l, we take forward pass for a training sample $\boldsymbol{x}_i$ to obtain the outputs $\boldsymbol{z}_i^{l-1}$ of $(l-1)$-th layer ($\boldsymbol{x}_i^l \equiv \boldsymbol{z}_i^{l-1}$ as input of l-th layer). For a convolution layer, we extract an input feature vector $\boldsymbol{r}_i^l$ by taking a patch vector from the 3-dimensional feature map $\boldsymbol{z}_i^{l-1}$; while for a linear layer, the input feature vector is simply $\boldsymbol{r}^l_i = \boldsymbol{z}^{l-1}_i$. We then concatenate all d-dimension input feature vectors along the column to construct an input-representation matrix $\mathbf{R}^l = [\boldsymbol{r}^l_1, \boldsymbol{r}^l_2, \cdots, \boldsymbol{r}^l_{n^*}] \in \mathbb{R}^{d \times n^*}$. Next, we can compute the basis vectors $\boldsymbol{U}^l$ and eigen-values $\boldsymbol{\Sigma}^l$ by using SVD as follows:

$$\boldsymbol{R}^l(\boldsymbol{R}^l)^\top = \sum_{\forall i} \boldsymbol{R}^l_i(\boldsymbol{R}^l_i)^\top = \boldsymbol{U}^l(\boldsymbol{\Sigma}^l)^2(\boldsymbol{U}^l)^\top, \qquad (2)$$

where $\boldsymbol{R}^l_i \in \mathbb{R}^{d \times m}$ is a subset of $\boldsymbol{R}^l$ including m input representations; hence, $\boldsymbol{R}^l(\boldsymbol{R}^l)^\top$ can be computed in mini-batch manner. As a result, $\boldsymbol{U}^l$ and $\boldsymbol{\Sigma}^l$ can be computed very efficiently (noting that d is usually small, e.g., < 1000).

Whenever a data deletion request is received for $\mathcal{D}_f$, we can compute the basis vectors of weight gradient space for the retaining dataset $\mathcal{D}_r$ as follows (note that $\boldsymbol{R}^l = [\boldsymbol{R}^l_r, \boldsymbol{R}^l_f]$):

$$\begin{aligned} \boldsymbol{R}^l_r(\boldsymbol{R}^l_r)^\top &= \boldsymbol{R}^l(\boldsymbol{R}^l)^\top - \boldsymbol{R}^l_f(\boldsymbol{R}^l_f)^\top \\ \boldsymbol{U}^l_r(\boldsymbol{\Sigma}^l_r)^2(\boldsymbol{U}^l_r)^\top &= \boldsymbol{U}^l(\boldsymbol{\Sigma}^l)^2(\boldsymbol{U}^l)^\top - \boldsymbol{R}^l_f(\boldsymbol{R}^l_f)^\top. \end{aligned} \qquad (3)$$

We pre-compute and cache the basis vectors $\boldsymbol{U}^l$ and eigen-values $\boldsymbol{\Sigma}^l$ of the input representations of each layer of the full training set, that allows us to efficiently compute the basis vectors and eigen-values of the retaining data, i.e., $\boldsymbol{U}^l_r$ and $\boldsymbol{\Sigma}^l_r$, when the forgetting dataset is given. Importantly, our method does not require the training data[1]; so it is applicable even when the training data is not accessible anymore (except for the forgetting data $\mathcal{D}_f$ that should be provided again by the users requesting data deletion). Our method can avoid the privacy concern over storing training data.

Next, we can obtain the Core Gradient space, $CGS = span\{\boldsymbol{u}^l_{r,1}, \boldsymbol{u}^l_{r,2}, \cdots, \boldsymbol{u}^l_{r,k}\}$, spanned by the first k vectors of $\boldsymbol{U}^l_r$, where k satisfies the following criteria for a given threshold γ^l: $\sum_{i=1}^k \sigma^l_{r,i} \geq \gamma^l \sum_{i=1}^d \sigma^l_{r,i}$ with $\boldsymbol{\Sigma}^l_r = \text{diag}([\sigma^l_{r,1}, \cdots, \sigma^l_{r,d}])$. CGS can be presented in matrix format as $\boldsymbol{M} = [\boldsymbol{u}^l_{r,1}, \boldsymbol{u}^l_{r,2}, \cdots, \boldsymbol{u}^l_{r,k}]$.

3.2.3 Gradient update

Given the forgetting dataset $\mathcal{D}_f$ and the loss function 1, we can compute the gradient $\nabla_{\mathbf{w}^l} L$. However, before applying the gradient step, the gradient $\nabla_{\mathbf{w}^l} L$ are first projected onto the CGS and then projected components are subtracted out from the gradient so that the remaining gradient components lie in the space orthogonal to CGS. The gradients are processed as follow:

$$\nabla_{\mathbf{w}^l} L_\perp = \nabla_{\mathbf{w}^l} L - \left(\nabla_{\mathbf{w}^l} L\right)\boldsymbol{M}^l(\boldsymbol{M}^l)^\top. \qquad (4)$$

For the context of de-poisoning application, since the unlearning of the forgetting dataset (i.e., the poisoned dataset) is strongly correlated with the learning of the retaining dataset (as we also aim to re-classify poisoned data), simply applying the naive orthogonal gradient projection (Eq. 4) will compromise the unlearning process, as noted in [27]. We address this issue by adopting the Trust Region Gradient Projection (TRGP) method [27][2] and using the CGS of the retaining dataset as the trust region. We apply TRGP only to the last linear layer in the de-poisoning application, while Eq. 4 is used in all other layers.

As with the training of a classification model, there is a potential for overfitting to occur during the unlearning process. To mitigate this issue, we adopt early stopping. Specifically, we compare the accuracy of the validation dataset to that of the forgetting dataset or to the random performance. By doing so, we can determine when the model starts to overfit to the unlearning and stop it early.

3.2.4 Incremental Unlearning

It is common for user data to be deleted multiple times during the life cycle of a model. Therefore, it is essential for the unlearning method to support incremental unlearning. This means that the method should be able to forget multiple batches of data one-by-one, allowing the model to adjust its weights accordingly as more training data is removed.

$$\begin{aligned} & \mathbf{w}^l_f = \mathbf{w}^l_o + \Delta\mathbf{w}^l_{o\to f} \\ \Rightarrow\ & \mathbf{w}^l_f \boldsymbol{r}^l_i = \mathbf{w}^l_o \boldsymbol{r}^l_i + \Delta\mathbf{w}^l_{o\to f} \boldsymbol{r}^l_i \\ \Rightarrow\ & \mathbf{w}^l_f \boldsymbol{r}^l_i \approx \mathbf{w}^l_o \boldsymbol{r}^l_i. \end{aligned} \qquad (5)$$

Here, we denote by $\mathbf{w}^l_o$ the weight of the originally model trained on $\mathcal{D}$ and by $\Delta\mathbf{w}^l_{o\to f}$ the update of weight from the original model to the unlearned model. Since our weight updates lie in the CGS of the retaining data, it implies that $\Delta\mathbf{w}^l_{o\to f}\boldsymbol{r}^l_i \approx 0$. As a result, the outputs of the unlearned model and those of the original model will be almost identical for the retaining dataset. Therefore, given the retaining dataset at any time, the basis vectors $\boldsymbol{U}^l_r$ and eigen-values $\boldsymbol{\Sigma}^l_r$ of the original model and unlearned model are the same, allowing us to easily re-compute the basis vectors whenever the retaining dataset changes (i.e., more data points are removed).

4. Experiments

4.1. Experiment setups

To evaluate the effectiveness of our proposed method on forgetting samples, we conduct experiments on two different datasets: CIFAR-10 [24] using AllCNN [42], SmallVGG (a small variation of VGG model [41] of 3 convolutional layers and 2 linear layers[3]) and TinyIma-

[1] When $\boldsymbol{V}^l$ is discarded, it is impossible to reconstruct $\boldsymbol{R}^l$.

[2] We refer readers to [27] for the details of TRGP method.

[3] More details about this model can be found in the source code.

Table 1. The experiment results for various readout functions for unlearning 500 samples of Class 0 of CIFAR10 using AllCNN model.

Method	Retrained	EUk	NTK	SCRUB	**PGU**
$\mathcal{D}_r$ error	0.04±0.01	0.06±0.01	3.49±0.98	0.01±0.00	0.05±0.01
$\mathcal{D}_f$ error	9.28±1.25	0.12±0.16	11.05±1.83	10.45±1.61	9.92±0.45
$\mathcal{D}_{\text{test}}$ error	9.78±0.19	10.13±0.15	12.68±0.74	10.26±0.44	9.86±0.15
Time (s)	1021±18	821±22	–*	321±43	88.45±4.12

* We do not report time for NTK since we can only samples a small subset of retaining dataset.

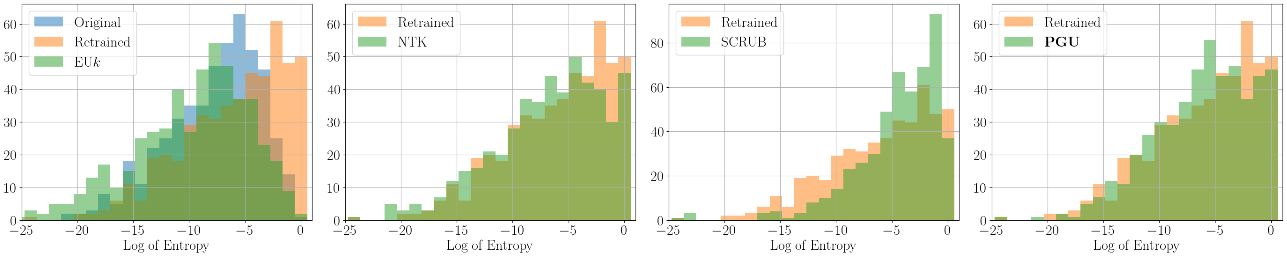


Figure 1. Distribution of the entropy of model output (confidence) of forgetting dataset $\mathcal{D}_f$ on original (before unlearning), retrained and unlearnt models using various methods.

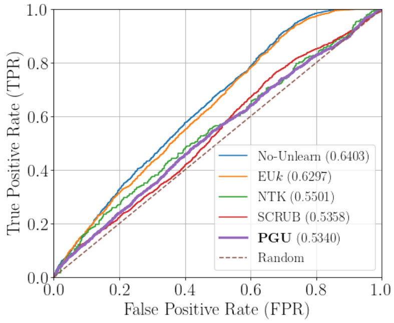


Figure 2. MIA ROC curve for various unlearnt models to unlearn 500 samples of CIFAR10 Class 0 using AllCNN model. The number in parenthesis is AUC.

geNet [46] using ResNet-18 [19]. The standard CIFAR-10/TinyImageNet consists of 50K/100K for training and 10K/10K for testing. In this paper, we split the standard 10K testing set into 5K-image validation set and 5K-image testing set. We train the models using Nesterov SGD for SmallVGG, AllCNN and Adam for ResNet-18 with starting learning rate at 0.01 and 0.001 respectively. During training, we utilize mini-batch size of 250 and adopt the exponential learning rate scheduler with the end learning rate set to 0.0005 after 200 epochs. We apply basic augmentation techniques, including random cropping and horizontal flipping.

For the unlearning process, we update only the model weights of convolutional and linear layers, and not the weights of batch-normalization layers or biases. We adopt the exponential learning rate scheduler with the starting learning rate of 0.05 and the ending learning rate of 0.01 after 100 epochs. For SmallVGG trained on CIFAR-10, we empirically choose the threshold $\gamma^l = 0.95$; for AllCNN trained on CIFAR-10, we empirically choose the threshold $\gamma^l = 0.9$; and for ResNet18 trained on TinyImageNet, we empirically choose the threshold $\gamma^l = 0.9$. We set $\lambda = 0.2$ for all experiments.

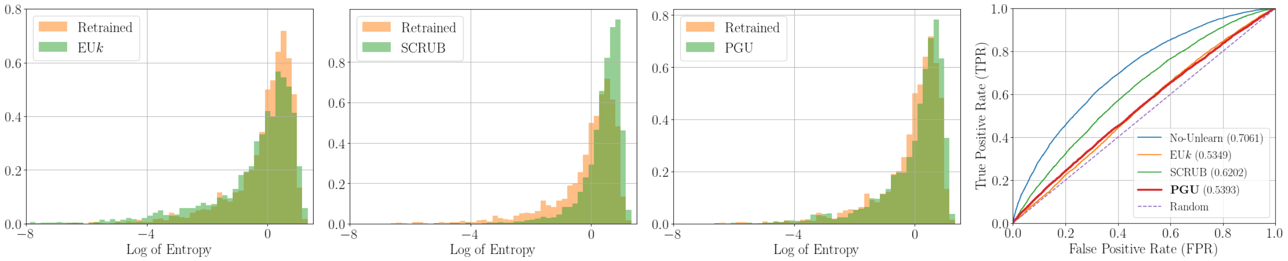
We use the following read-out functions, which may be used to gauge how much information they were able to destroy: **(i) Error on the test set** $\mathcal{D}_{\text{test}}$: ideally small, **(ii) Error on the subset to be forgotten** $\mathcal{D}_f$: ideally the same as the error of a model trained without seeing $\mathcal{D}_f$, **(iii) Error on the retaining set** $\mathcal{D}_r$: ideally no change after the unlearn process or similar to that of the retrained model, **(iv) Membership inference attack (MIA):** To conduct a membership inference attack, we train 2 sets of models: 50 original models with full training data and 50 retrained models with only the retaining dataset. We then treat the attack as a binary classification problem on model outputs of the forget dataset, where class **1** represents the samples seen during training (i.e., from original models) and class **0** represents the samples not seen during training (i.e., from the retrained model). For this classification, we use 40 models to extract a training set, 5 models to extract a validation set, and 5 models to extract a test set. We train a XGBoost classifier and finetune to achieve the best F1 score on the validation set. We visualize the Receiver Operating Characteristic (ROC) curve and report the Area Under Curve (AUC). The ROC curve shows trade-off between true-positive rate (TPR) and false-positive rate (FPR)[4]. Furthermore, it is worth noting that our assumption about the knowledge of attacker is quite strong; specifically, we assume that the attacker knows the model architecture, training methods, the full labelled training and forgetting dataset. Employing such a strong as-

[4] Please refer to Supplementary for more details of the MIA setting.

Table 2. The experiment results for various readout functions for unlearning classes using ResNet-18 model trained on TinyImageNet.

	Method	Original	Retrained	EUk	SCRUB	**PGU**
5 Classes	$\mathcal{D}_{\text{retain test}}{}^*$ error	52.78±0.36	52.58±0.17	53.15±0.28	51.80±0.37	53.03±0.43
	$\mathcal{D}_r$ error	0.56±0.04	0.56±0.05	1.84±0.08	0.20±0.00	0.74±0.06
	Time (s)	–	3,846±118	2318±97	1693±65	765±45

* $\mathcal{D}_{\text{test}}$ after excluding samples of forgetting classes.



(a) (b) (c) (d)

Figure 3. Distribution of the entropy of model output (confidence) of forgetting dataset $\mathcal{D}_f$ on retrained and unlearnt models using various methods (Fig a, b, and c) and MIA ROC curve (Fig d) when forgetting 5 classes of TinyImageNet using ResNet-18.

sumption simplifies the task of quantifying the extent to which information from the forgetting set still retains within the model. This might be more challenging in a more realistic scenario due to the biases and variations caused by shallow models, shallow training-testing sets, etc. **(v) Model confidence**: we visualize the distribution of model confidence (entropy of the output prediction) on the forget set $\mathcal{D}_f$, **(vi) Unlearning time:** should be significantly smaller than the retraining time.

To obtain reliable results, we conduct each experiment 5 times and report the mean and standard deviation, except for MIA experiment which is executed only one. We compare our method PGU with recent works: NTK[5] [1], Exact Unlearning-k (EU-k)[6] [13], and SCRUB [25]. Importantly, we note that our method only requires the forgetting dataset and is applicable even when the training data is no longer accessible. The following experiment results demonstrate that our unlearnt method can achieve favorable results across various readout functions in comparison with other works which requires retaining dataset. Furthermore, our unlearnt models are closely matched with retrained models, which are the optimal targets for unlearning.

4.2. Data removal

First, we conduct experiment to selective remove some training data. Specifically, we remove 500 samples of the first class (Class 0) of CIFAR10 dataset using AllCNN.

The experimental results, as presented in Table 1, show the favorable performance of our proposed method in comparison to alternative approaches. Remarkably, our unlearnt model closely matches with the retrained model even without using the retaining dataset. Specifically, our method exhibits the smallest increase in error on $\mathcal{D}_{\text{test}}$. Notably, we found that EUk method [13] performs poorly on many readout functions, i.e., $\mathcal{D}_f$ error is very small and similar to $\mathcal{D}_r$ error. This observation suggests that even though the last few layers are retrained from scratch, information from the forgetting dataset can still be leaked from shallower layers. We observe a larger increases in $\mathcal{D}_r$ $\mathcal{D}_{\text{test}}$ errors for NTK methods; this is potentially because we can only sample 1500 samples (out of 49,500 samples) as the retaining dataset due to its scalability limitation.

Considering model confidence, as depicted in Figure 1, it is evident that our unlearnt method produces a $\mathcal{D}_f$ entropy distribution which aligns with that of the retrained model the best. NTK, despite of its scalability limitation, can produce a $\mathcal{D}_f$ entropy distribution which also closely aligns with that of retrained model. Furthermore, the $\mathcal{D}_f$ entropy distribution of EUk is closer the the $\mathcal{D}_f$ entropy distribution of original model than that of retrain model. This outcome corroborates the findings from Table 1.

Figure 2 demonstrates the effectiveness of our approach in erasing information related to forgotten datasets from the model. Specifically, there is a substantial decrease in the success rate of MIA attackers, with the Area Under Curve (AUC) dropping from 0.6403 to 0.5340 (slightly higher than random chances). In term of AUC for MIA ROC curve, our method achieves comparable results to SCRUB [25], showcasing its merits as it does not require the retaining dataset. Finally, our approach outpaces both the retrained method and alternative techniques in terms of speed. Its efficiency comes from the fact that it solely relies on the forgetting dataset, usually at a small size; whereas alternative methods need to fine-tuning on the large retaining dataset. It is worth mentioning that the training time of our approach

[5] Due to the scalability issue of the method, we can only select 1500 samples as the retaining dataset in calculation.

[6] We retrain the last 3 convolutional/linear layers.

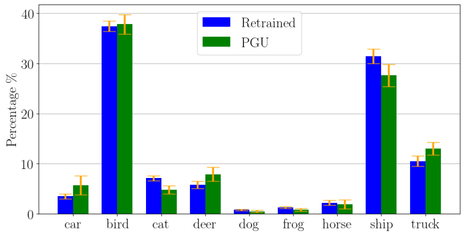


Figure 4. How retrained and our unlearnt models classify forgetting samples (airplane).

includes the SVD computation time, specifically SVD computation consumes less than 6 seconds in this experiment.

4.3. Class removals

The above setting shows applications where samples are randomly removed. Another appealing application of unlearning involves completely removing samples from specific classes. To test the effectiveness of our proposed method in this application, we conducted experiments using ResNet-18 on TinyImageNet dataset and SmallVGG on CIFAR-10 dataset. Without losing generality, we select first 5 classes of TinyImageNet and the first class of CIFAR-10 as the forgetting set. Noting that for this class removal task, we remove the rows corresponding to forgetting classes in the last linear layer of unlearnt models before evaluating.

In Table 2, we present the outcomes of our experiments with different readout functions. We observe that classification errors of the our unlearnt model on retaining training and retaining testing set (composed of the training and testing sets excluding the samples from the forgetting class) are more similar to those of the retrained models as compared to EUk, SCRUB. The $\mathcal{D}_f$ entropy distribution of our unlearnt model also aligns with that of the retrained model better than EUk, SCRUB (Figure 3a, b, c). In term of MIA when forgetting 5 classes, our method can achieve better AUC than that of SCRUB while only being slight lower than that of EUk (Figure 3d). Finally, our method is faster than other methods and significantly faster than the retrained approach. In summary, our comprehensive analysis across various metrics underscores the advantages offered by our proposed method.

We further analyse the impact of unlearning a class on retaining classes using SmallVGG on CIFAR-10 and unlearning the first class (i.e., airplane). Figure 4 depicts how (percentage % of samples) the unlearnt and retrained models will classify the forgotten samples in CIFAR-10 dataset. The figure shows that the unlearnt and retrained models exhibit similar behaviors when presented airplane samples. Specifically, we can see that both models will more likely to classify an airplane sample as bird or ship, potentially due to the similarity in the background of those classes. Conversely, both models are also less likely to classify an airplane as a dog, a frog, or a horse.

4.4. Incremental data removals

We performed sequential unlearning experiments where we unlearnt a subset of 1,000 training samples at a time. In the first run, we unlearnt a set of 1,000 samples, and the resulting unlearnt model was used to subsequently unlearn another subset of 1,000 training samples. The subset of 1,000 forget samples in each run are formed by randomly selecting 100 samples for each class in the remaining training samples.

Table 3 demonstrates that the unlearnt model achieves very similar test errors to the retrained models, while only minor increases in error rate for retaining set $\mathcal{D}_r$ and forgetting set $\mathcal{D}_f$ are observed. In terms of MIA success rate, Figure 6 shows that our method results in the ROC curve that closely aligns with the random line. This suggests that attackers are unable to determine whether a forgotten sample was employed in training process any more effectively than random guessing.

Additionally, Figure 5 shows the distributions of log-entropy of the unlearnt and retrained models on the forgetting dataset $\mathcal{D}_f$ before and after unlearning. We can observe that after unlearning the log-entropy distributions of the unlearnt and retrained models are closely matched to each other. Figure 5 and Table 3 indicate that our method can support incremental unlearning without significantly affecting performance of test set $\mathcal{D}_{\text{test}}$. Finally, our method is significantly faster than the retrained approach.

4.5. Depoisoning

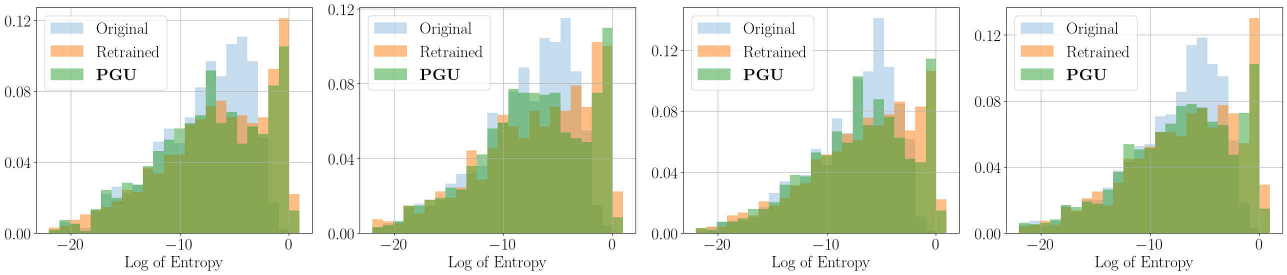
In this section, we focus on the poisoning attack setting on machine learning. Specifically, an adversary aims at misleading a model by flipping labels of the training data. The label flips significantly degrade the performance of the learning model. Hence, we use the unlearning method to remove the harmful effect of poisoned samples and correct the model to achieve the performance as if the model was trained without the poisoned samples.

For this experiment, we use the SmallVGG model on CIFAR10 dataset. For fair comparison to [44], we follow their experimental setting. Specifically, we change the labels of certain pairs of classes by flipping a portion of them to their counterpart. For example, a sample with the label ”cat” could be changed to ”dog” and vice versa. This type of attack can result in similar performance degradation as other label-flip attacks [44]. We evaluate our unlearning method on this setting with different poisoning budgets.

The experimental results are presented in Table 4. These results demonstrate that our unlearning method can effectively mitigate the detrimental effects of poisoned samples on the model, resulting in significant performance recovery across different levels of poisoning (including cases where up to 50% of the samples are poisoned). Additionally, our method also significantly surpasses the method in [44] in

Table 3. The experiment results for various readout functions when conducting incremental unlearning on SmallVGG trained on CIFAR10.

	Num. Forget	0 - 1K	1K - 2K	2K - 3K	3K - 4K
Retrained	$\mathcal{D}_{\text{test}}$ error (%)	9.62±0.52	9.55±0.44	9.38±0.32	9.84±0.43
	$\mathcal{D}_r$ error (%)	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00
	$\mathcal{D}_f$ error (%)	9.61±0.53	9.51±0.43	9.59±0.30	9.69±0.26
	Time (s)	1030±19	1020±20	1009±21	995±22
PGU	$\mathcal{D}_{\text{test}}$ error (%)	9.84±1.08	9.86±0.99	9.78±0.95	9.80±1.11
	$\mathcal{D}_r$ error (%)	0.00±0.00	0.00±0.00	0.01±0.00	0.01±0.00
	$\mathcal{D}_f$ error (%)	10.02±0.97	9.93±1.10	10.02±1.24	11.30±1.37
	Time (s)	156±21	161±19	178±29	172±24



(a) 0 - 1K (b) 1K - 2K (c) 2K - 3K (d) 3K - 4K

Figure 5. Distribution of the entropy of model output (confidence) of forgetting dataset $\mathcal{D}_f$ on original (before unlearning), retrained and our unlearnt models.

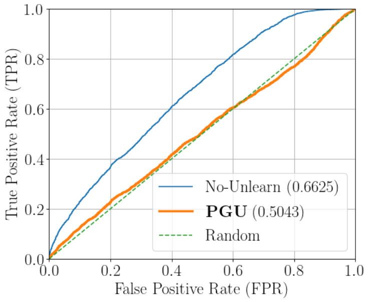


Figure 6. MIA ROC curve after incremental unlearning 4K samples (1K step size) of CIFAR10 on SmallVGG model.

Table 4. Test accuracy for different poisoning budgets for SmallVGG trained CIFAR10. The SmallVGG model achieves 90.16% in test accuracy with *full-clean* training data. We present the accuracy of the model trained with poisoned dataset (***Poisoned***), the model trained with clean data only (exclude the poisoned samples) (***Clean***), and the poisoned model after depoisoning using PGU and [44].

Num. Poison	5K	10K	15K	20K	25K
Poisoned	86.07	79.32	69.69	56.78	45.36
Clean	89.81	88.89	88.69	87.68	86.86
PGU	**87.95**	**86.87**	**85.46**	**85.13**	**83.12**
1st Order [44]	86.56	83.77	79.68	74.11	67.98
2nd Order [44]	87.43	84.13	79.80	74.28	68.28

term of performance recovery, especially when a large number of poisoned samples are presented. Importantly, our approach does not require access to the clean labels of the poisoned samples.

5. Conclusion

We introduce a new machine unlearning method to remove the effect of a specific subset of training data on the trained model. It has important applications in ensuring the "right to be forgotten" in the context of user privacy, erasing a subset of malicious or adversarial data from the model. The proposed method shows promising empirical performance for different model architectures and datasets across various readout functions. We also show the ability to approximately unlearn for large models very efficiently without any additional limitations beyond those encountered during training. Additionally, as not requiring the retaining dataset, our proposed method can be effectively employed when the training data is no longer accessible, a scenario where the majority of existing approaches are not applicable. For future work, we plan to extend this approach to other learning applications beyond classification. Additionally, we aim to test the effectiveness of our method against advanced Membership Inference Attack and model inversion techniques, as these are active research areas.

Acknowledgement

This research was partially funded by the Australian Government through the Australian Research Council (ARC). Prof Venkatesh is the recipient of an ARC Australian Laureate Fellowship (FL170100006).

References

[1] Aditya Golatkar and Alessandro Achille and Stefano Soatto. Eternal Sunshine of the Spotless Net: Selective Forgetting in Deep Networks. In *CVPR*, pages 9301–9309, 2020. 1, 2, 6

[2] Galen Andrew, Om Thakkar, Hugh Brendan McMahan, and Swaroop Ramaswamy. Differentially private learning with adaptive clipping. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. 3

[3] Thomas Baumhauer, Pascal Schöttle, and Matthias Zeppelzauer. Machine unlearning: Linear filtration for logit-based classifiers. 111(9):3203–3226, 2022. 1, 2

[4] Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 141–159, 2021. 2

[5] Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *2015 IEEE Symposium on Security and Privacy*, pages 463–480, 2015. 1, 2

[6] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. Membership inference attacks from first principles. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1897–1914. IEEE, 2022. 3

[7] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 267–284, Aug. 2019. 1

[8] Christopher A Choquette-Choo, Florian Tramer, Nicholas Carlini, and Nicolas Papernot. Label-only membership inference attacks. In *International conference on machine learning*, pages 1964–1974, 2021. 3

[9] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography*, pages 265–284, 2006. 3

[10] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3–4):211–407, aug 2014. 3

[11] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, page 1322–1333, 2015. 1

[12] Shaopeng Fu, Fengxiang He, and Dacheng Tao. Knowledge removal in sampling-based bayesian inference. In *International Conference on Learning Representations*, 2022. 1

[13] Shashwat Goel, Ameya Prabhu, and Ponnurangam Kumaraguru. Evaluating inexact unlearning requires revisiting forgetting, 01 2022. 6

[14] A. Golatkar, A. Achille, A. Ravichandran, M. Polito, and S. Soatto. Mixed-privacy forgetting in deep networks. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 792–801, jun 2021. 1

[15] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Forgetting Outside the Box: Scrubbing Deep Networks of Information Accessible from Input-Output Observations. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 383–398, 2020. 1, 2

[16] Aditya Golatkar, Alessandro Achille, Yu-Xiang Wang, Aaron Roth, Michael Kearns, and Stefano Soatto. Mixed differential privacy in computer vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8376–8386, June 2022. 3

[17] Sivakanth Gopi, Yin Tat Lee, and Lukas Wutschitz. Numerical composition of differential privacy. In *NeurIPS 2021*, June 2021. 3

[18] Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. Certified data removal from machine learning models. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20, 2020. 1, 2

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016. 5

[20] Zachary Izzo, Mary Anne Smart, Kamalika Chaudhuri, and James Zou. Approximate data deletion from machine learning models. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 2008–2016, 13–15 Apr 2021. 1

[21] Masayuki Karasuyama and Ichiro Takeuchi. Multiple incremental decremental learning of support vector machines. *IEEE Transactions on Neural Networks*, 21(7):1048–1059, 2010. 2

[22] Junyaup Kim and Simon S. Woo. Efficient two-stage model retraining for machine unlearning. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 4360–4368, 2022. 1

[23] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1885–1894, 2017. 1, 2

[24] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. 4

[25] Meghdad Kurmanji, Peter Triantafillou, and Eleni Triantafillou. Towards unbounded machine unlearning, 02 2023. 2, 6

[26] Jaehoon Lee, Lechao Xiao, Samuel S. Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 2019. 2

[27] Sen Lin, Li Yang, Deliang Fan, and Junshan Zhang. TRGP: Trust Region Gradient Projection for Continual Learning. 2022. 2, 3, 4

[28] A. Mantelero. The EU proposal for a general data protection regulation and the roots of the 'right to be forgotten'. In *Computer Law Security Review*, page 29(3):229–235, 2013. 1

[29] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM Comput. Surv.*, 54(6), jul 2021. 1

[30] Ronak Mehta, Sourav Pal, Vikas Singh, and Sathya N. Ravi. Deep unlearning via randomized conditionally independent hessians. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10422–10431, June 2022. 1, 2

[31] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE symposium on security and privacy (SP)*, pages 691–706. IEEE, 2019. 1

[32] Quoc Phong Nguyen, Bryan Kian, Hsiang Low, and Patrick Jaillet. Variational bayesian unlearning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, 2020. 1

[33] Quoc Phong Nguyen, Ryutaro Oikawa, Dinil Mon Divakaran, Mun Choon Chan, and Bryan Kian Hsiang Low. Markov chain monte carlo-based machine unlearning: Unlearning what needs to be forgotten. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '22, page 351–363, 2022. 1

[34] Stuart L. Pardau. THE CALIFORNIA CONSUMER PRIVACY ACT: TOWARDS A EUROPEAN-STYLE PRIVACY REGIME IN THE UNITED STATES? In *Journal of Technology Law Policy*, volume 23, 2018. 1

[35] Enrique Romero, Ignacio Barrio, and Lluís Belanche. Incremental and decremental learning for linear support vector machines. In Joaquim Marques de Sá, Luís A. Alexandre, Włodzisław Duch, and Danilo Mandic, editors, *Artificial Neural Networks – ICANN 2007*, pages 209–218, 2007. 2

[36] Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, Yann Ollivier, and Hervé Jégou. White-box vs black-box: Bayes optimal strategies for membership inference. In *International Conference on Machine Learning*, 2019. 3

[37] Gobinda Saha, Isha Garg, Aayush Ankit, and Kaushik Roy. Space: Structured compression and sharing of representational space for continual learning. *IEEE Access*, 9:150480–150494, 2021. 2, 3

[38] Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. In *International Conference on Learning Representations*, 2021. 2, 3

[39] Ayush Sekhari, Jayadev Acharya, Gautam Kamath, and Ananda Theertha Suresh. Remember what you want to forget: Algorithms for machine unlearning. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. 2

[40] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18, Los Alamitos, CA, USA, may 2017. IEEE Computer Society. 1, 3

[41] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 4

[42] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for Simplicity: The All Convolutional Net. In *ICLR Workshop*, 2015. 4

[43] 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 (EuroSamp;P)*, pages 303–319, jun 2022. 2

[44] Alexander Warnecke, Lukas Pirch, Christian Wressnegger, and Konrad Rieck. Machine Unlearning of Features and Labels. *Network and Distributed System Security (NDSS)*, 2023. 1, 7, 8

[45] Xi Wu, Matthew Fredrikson, Somesh Jha, and Jeffrey F. Naughton. A methodology for formalizing model-inversion attacks. In *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*, pages 355–370, 2016. 1

[46] Ya Le; Xuan S. Yang. Tiny imagenet visual recognition challenge. Technical report, 2015. 5

[47] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017. 3

[48] Zijie Zhang, Yang Zhou, Xin Zhao, Tianshi Che, and Lingjuan Lyu. Prompt certified machine unlearning with randomized gradient smoothing and quantization. In *Advances in Neural Information Processing Systems*, pages 13433–13455, 2022. 2