Eternal Sunshine of the Spotless Net: Selective Forgetting in Deep Networks

Aditya Golatkar
UCLA
aditya29@cs.ucla.edu

Alessandro Achille
UCLA
achille@cs.ucla.edu

Stefano Soatto
UCLA
soatto@ucla.edu

Abstract

We explore the problem of selectively forgetting a particular subset of the data used for training a deep neural network. While the effects of the data to be forgotten can be hidden from the output of the network, insights may still be gleaned by probing deep into its weights. We propose a method for "scrubbing" the weights clean of information about a particular set of training data. The method does not require retraining from scratch, nor access to the data originally used for training. Instead, the weights are modified so that any probing function of the weights is indistinguishable from the same function applied to the weights of a network trained without the data to be forgotten. This condition is a generalized and weaker form of Differential Privacy. Exploiting ideas related to the stability of stochastic gradient descent, we introduce an upper-bound on the amount of information remaining in the weights, which can be estimated efficiently even for deep neural networks.

1. Introduction

Say you are the number '6' in the MNIST handwritten digit database. You are proud of having nurtured the development of convolutional neural networks and their many beneficial uses. But you are beginning to feel uncomfortable with the attention surrounding the new "AI Revolution," and long to not be recognized everywhere you appear. You wish a service existed, like that offered by the firm Lacuna INC in the screenplay *The Eternal Sunshine of the Spotless Mind*, whereby you could submit your images to have your identity scrubbed clean from handwritten digit recognition systems. Before you, the number '9' already demanded that digit recognition systems returned, instead of a ten-dimensional "pre-softmax" vector (meant to approximate the log-likelihood of an image containing a number from 0 to 9) a *nine*-dimensional vector that excluded the number '9'. So now, every image showing '9' yields an outcome at random between 0 and 8. Is this enough? It could be that the system still contains *information* about the number '9,' and just suppresses it in the output. How do you know that the system has truly forgotten about you, even *inside the black box*? Is it possible to scrub the system so clean that it behaves as if it had never seen an image of you? Is it possible to do so without sabotaging information about other digits, who wish to continue enjoying their celebrity status? In the next section we formalize these questions to address the problem of *selective forgetting in deep neural networks* (DNNs). Before doing so, we present a summary of our contributions in the context of related work.

1.1. Related Work

Tampering with a learned model to achieve, or avoid, forgetting pertains to the general field of *life-long learning*. Specifically for the case of deep learning and representation learning, this topic has algorithmic, architectural and modeling ramifications, which we address in order.

Differential privacy [8] focuses on guaranteeing that the parameters of a trained model do not leak information about *any* particular individual. While this may be relevant in some applications, the condition is often too difficult to enforce in deep learning (although see [1]), and not always necessary. It requires the possible distribution of weights, given the dataset, $P(w|\mathcal{D})$ to remain almost unchanged after replacing a sample. Our definition of selective forgetting can be seen as a generalization of differential privacy. In particular, we do not require that information about *any* sample in the dataset is minimized, but rather about a particular subset $\mathcal{D}_f$ selected by the user. Moreover, we can apply a "scrubbing" function $S(w)$ that can perturb the weights in order to remove information, so that $P(S(w)|\mathcal{D})$, rather than $P(w|\mathcal{D})$, needs to remain unchanged. This less restrictive setting allows us to train standard deep neural networks using stochastic gradient descent (SGD), while still being able to ensure forgetting.

Deep Neural Networks can memorize details about particular instances, rather than only shared characteristics [29, 4]. This makes forgetting critical, as attackers can try to extract information from the weights of the model. **Membership attacks** [28, 15, 24, 14, 26] attempt to determine whether a particular cohort of data was used for training, without any constructive indication on how to actively forget it. They relate to the ability of **recovering data from**

the model [10] which exploits the increased confidence of the model on the training data to reconstruct images used for training; [23] proposes a method for performing zero-shot knowledge distillation by adversarially generating a set of exciting images to train a student network. [25] proposes a definition of forgetting based on changes of the value of the loss function. We show that this is not meaningful forgetting, and in some cases it may lead to the (opposite) "Streisand effect," where the sample to be forgotten is actually made more noticeable.

Stability of SGD. In [13], a bound is derived on the divergence of training path of models trained with the same random seed (*i.e.*, same initialization and sampling order) on datasets that differ by one sample (the "stability" of the training path). This can be considered as a measure of memorization of a sample and, thus, used to bound the generalization error. While these bounds are often loose, we introduce a novel bound on the residual information about a set of samples to be forgotten, which exploits ideas from both the stability bounds and the PAC-Bayes bounds [22], which have been successful even for DNNs [9].

Machine Unlearning was first studied by [7] in the context of statistical query learning. [5] proposed an unlearning method based on dataset sharding and training multiple models. [11] proposed an efficient data elimination algorithm for k-means clustering. However, none of these methods can be applied for deep networks. The term "forgetting" is also used frequently in life-long learning, but often with different connotations that in our work: **Catastrophic forgetting**, where a network trained on a task rapidly loses accuracy on that task when fine-tuned for another. But while the network can forget a *task,* the information on the *data* it used may still be accessible from the weights. Hence, even catastrophic forgetting does not satisfy our stronger definition. Interestingly, however, our proposed solution for forgetting relates to techniques used to *avoid* forgetting: [17] suggests adding an L_2 regularizer using the Fisher Information Matrix (FIM) of the task. We use the FIM, restricted to the samples we wish to retain, to compute the optimal noise to destroy information, so that a cohort can be forgotten while maintaining good accuracy for the remaining samples. Part of our forgetting algorithm can be interpreted as performing "optimal brain damage" [19] in order to remove information from the weights if it is useful only or mainly to the class to be forgotten.

In this paper we talk about the weights of a network as containing "information," even though we have *one* set of weights whereas information is commonly defined only for random variables. While this has caused some confusion in the literature, [3] proposes a viable formalization of the notion, which is compatible with our framework. Thus, we will use the term "information" liberally even when talking about a particular dataset and set of weights.

In defining forgetting, we wish to be resistant to both "black-box" attacks, which only have access to the model output through some function (API), and "white-box" attacks, where the attacker can additionally access the model weights. Since at this point it is unclear how much information about a model can be recovered by looking only at its inputs and outputs, to avoid unforeseen weaknesses we characterize forgetting for the stronger case of white-box attacks, and derive bounds and defense mechanism for it.

1.2. Contributions

In summary, our contributions are, first, to propose a **definition of selective forgetting** for trained neural network models. It is not as simple as obfuscating the activations, and not as restrictive as Differential Privacy. Second, we propose a **scrubbing procedure** that removes information from the trained weights, without the need to access the original training data, nor to re-train the entire network. We compare the scrubbed network to the gold-standard model(s) trained from scratch without any knowledge of the data to be forgotten. We also prove the optimality of this procedure in the quadratic case. The approach is applicable to both the case where an entire class needs to be forgotten (e.g. the number '6') or multiple classes (e.g., all odd numbers), or a particular subset of samples within a class, while still maintaining output knowledge of that class. Our approach is applicable to networks pre-trained using standard loss functions, such as cross-entropy, unlike Differential Privacy methods that require the training to be conducted in a special manner. Third, we introduce a **computable upper bound** to the amount of the retained information, which can be efficiently computed even for DNNs. We further characterize the optimal tradeoff with preserving complementary information. We illustrate the criteria using the MNIST and CIFAR-10 datasets, in addition to a new dataset called "Lacuna."

1.3. Preliminaries and Notation

Let $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$ be a dataset of images x_i, each with an associated label $y_i \in \{1, \dots, K\}$ representing a class (or label, or identity). We assume that $(x_i, y_i) \sim P(x, y)$ are drawn from an unknown distribution P.

Let $\mathcal{D}_f \subset \mathcal{D}$ be a subset of the data (cohort), whose information we want to remove (scrub) from a trained model, and let its complement $\mathcal{D}_r := \mathcal{D}_f^\complement$ be the data that we want to retain. The data to forget $\mathcal{D}_f$ can be any subset of $\mathcal{D}$, but we are especially interested in the case where $\mathcal{D}_f$ consists of all the data with a given label k (that is, we want to completely forget about a class), or a subset of a class.

Let $\phi_w(\cdot) \in \mathbb{R}^K$ be a parametric function (model), for instance a DNN, with parameters w (weights) trained using $\mathcal{D}$ so that the k-th component of the vector ϕ_w in response to an image x approximates the optimal discriminant (log-

posterior), $\phi_w(x)_k \simeq \log P(y=k|x)$, up to a normalizing constant.

1.4. Training algorithm and distribution of weights

Given a dataset $\mathcal{D}$, we can train a model — or equivalently a set of weights — w using some training algorithm A, that is $w = A(\mathcal{D})$, where $A(\mathcal{D})$ can be a stochastic function corresponding to a stochastic algorithm, for example stochastic gradient descent (SGD). Let $P(w|\mathcal{D})$ denote the distribution of possible outputs of algorithm A, where $P(w|\mathcal{D})$ will be a degenerate Dirac delta if A is deterministic. The scrubbing function $S(w)$ — introduced in the next section — is also a stochastic function applied to the weights of a trained network. We denote by $P(S(w)|\mathcal{D})$ the distribution of possible weights obtained after training on the dataset $\mathcal{D}$ using algorithm A and then applying the scrubbing function $S(w)$. Given two distributions $p(x)$ and $q(x)$, their Kullback-Leibler (KL) divergence is defined by $\mathrm{KL}\left(p(x)\,\|\,q(x)\right) := \mathbb{E}_{x\sim p(x)}\left[\log\left(p(x)/q(x)\right)\right]$. The KL-divergence is always positive and can be thought of as a measure of similarity between distributions. In particular it is zero if and only if $p(x) = q(x)$. Given two random variables x and y, the amount of Shannon Mutual Information that x has about y is defined as $I(x;y) := \mathbb{E}_x\left[\mathrm{KL}\left(p(y|x)\,\|\,p(y)\right)\right]$.

2. Definition and Testing of Forgetting

Let ϕ_w be a model trained on a dataset $\mathcal{D} = \mathcal{D}_f \sqcup \mathcal{D}_r$ Then, a forgetting (or "scrubbing") procedure consists in applying a function $\mathrm{S}(w;\mathcal{D}_f)$ to the weights, with the goal of forgetting, that is to ensure that an "attacker" (algorithm) in possession of the model ϕ_w cannot compute some "readout function" $f(w)$, to reconstruct information about $\mathcal{D}_f$.

It should be noted that one can always infer *some* properties of $\mathcal{D}_f$, even without having ever seen it. For example, if $\mathcal{D}$ consists of images of faces, we can infer that images in $\mathcal{D}_f$ are likely to display two eyes, even without looking at the model w. What matters for forgetting is the amount of *additional* information $f(w)$ can extract from a cohort $\mathcal{D}_f$ by exploiting the weights w, that could not have been inferred simply by its complement $\mathcal{D}_r$. This can be formalized as follows:

Definition 1. Given a readout function f, an optimal scrubbing function for f is a function $\mathrm{S}(w;\mathcal{D}_f)$ — or $\mathrm{S}(w)$, omitting the argument $\mathcal{D}_f$ — such that there is another function $\mathrm{S}_0(w)$ that does not depend on $\mathcal{D}_f$[1] for which:

$$\mathrm{KL}\left(P(f(\mathrm{S}(w))|\mathcal{D})\,\|\,P(f(\mathrm{S}_0(w))|\mathcal{D}_r)\right) = 0. \quad (1)$$

[1] If S_0 could depend on $\mathcal{D}_f$, we could take $S(w) = w$ to be the identity, and let $S_0(w)$ ignore w and obtain new weights by training from scratch on $\mathcal{D}$ — that is $S_0(w) = w'$ with $w' \sim p(w|\mathcal{D})$. This brings the KL to zero, but does not scrub any information, since $S(w)$ is the identity.

The function $S_0(w)$ in the definition can be thought as a *certificate* of forgetting, which shows that $S(w)$ is indistinguishable from a model that has never seen $\mathcal{D}_f$. Satisfying the condition above is trivial by itself, *e.g.*, by choosing $S(w) = S_0(w) = c$ to be constant. The point is to do so while retaining as much information as possible about $\mathcal{D}_r$, as we will see later when we introduce the Forgetting Lagrangian in eq. (4). The formal connection between this definition and the amount of Shannon Information about $\mathcal{D}_f$ that a readout function can extract is given by the following:

Proposition 1. Let the forgetting set $\mathcal{D}_f$ be a random variable, for instance, a random sampling of the data to forget. Let Y be an attribute of interest that depends on $\mathcal{D}_f$. Then,

$$I(Y; f(S(w))) \leq \mathbb{E}_{\mathcal{D}_f}[\mathrm{KL}\left(P(f(\mathrm{S}(w))|\mathcal{D})\,\|\,P(f(\mathrm{S}_0(w))|\mathcal{D}_r)\right)]. \quad (2)$$

Yet another interpretation of eq. (1) arises from noticing that, if that quantity is zero then, given the output of the readout function $f(w)$, we cannot predict with better-than-chance accuracy whether the model $w' = \mathrm{S}(w)$ was trained with or without the data. In other words, after forgetting, membership attacks will fail.

In general, we may not know what readout function a potential attacker will use, and hence we want to be robust to every $f(w)$. The following lemma is useful to this effect:

Lemma 1. For any function $f(w)$ we have:

$$\mathrm{KL}\left(P(f(\mathrm{S}(w))|\mathcal{D})\,\|\,P(f(\mathrm{S}_0(w))|\mathcal{D}_r)\right) \leq \mathrm{KL}\left(P(\mathrm{S}(w)|\mathcal{D})\,\|\,P(\mathrm{S}_0(w)|\mathcal{D}_r)\right).$$

Therefore, we can focus on minimizing the quantity

$$\mathrm{KL}\left(P(\mathrm{S}(w)|\mathcal{D})\,\|\,P(\mathrm{S}_0(w)|\mathcal{D}_r)\right), \quad (3)$$

which guarantees robustness to any readout function. For the sake of concreteness, we now give a first simple example of a possible scrubbing procedure.

Example 1 (Forgetting by adding noise)**.** Assume the weights w of the model are bounded. Let $S(w) = S_0(w) = w + \sigma n$, where $n \sim \mathcal{N}(0, I)$, be the scrubbing procedure that adds noise sampled from a Gaussian distribution. Then, as the variance σ increases, we achieve total forgetting:

$$\mathrm{KL}\left(P(\mathrm{S}(w)|\mathcal{D})\,\|\,P(\mathrm{S}_0(w)|\mathcal{D}_r)\right) \xrightarrow{\sigma\to\infty} 0.$$

While adding noise with a large variance does indeed help forgetting, it throws away the baby along with the bath water, rendering the model useless. Instead, we want to forget as much as possible about a cohort while retaining the accuracy of the model. This can be formalized by minimizing the **Forgetting Lagrangian**:

$$\mathcal{L} = \mathbb{E}_{S(w)}\left[L_{\mathcal{D}_r}(w)\right] + \lambda\,\mathrm{KL}\left(P(\mathrm{S}(w)|\mathcal{D})\,\|\,P(\mathrm{S}_0(w)|\mathcal{D}_r)\right), \quad (4)$$

where $L_{\mathcal{D}_r}(w)$ denotes the loss of the model w on the retained data $\mathcal{D}_r$. Optimizing this first term is relatively easy. The problem is doing so while also minimizing the second (forgetting) term: For a DNN, the distribution $P(w|\mathcal{D})$ of possible outcomes of the training process is complex making difficult the estimation of the KL divergence above, a problem we address in the next section. Nonetheless, the Forgetting Lagrangian, if optimized, captures the notion of **selective forgetting** at the core of this work.

2.1. Stability and Local Forgetting Bound

Given a stochastic training algorithm $A(\mathcal{D})$, we can make the dependency on the random seed ϵ explicit by writing $A(\mathcal{D}, \epsilon)$, where we assume that $A(\mathcal{D}, \epsilon)$ is now a deterministic function of the data and the random seed. We now make the following assumptions: (1) the cohort to be forgotten, $\mathcal{D}_f$, is a small portion of the overall dataset $\mathcal{D}$, lest one is better-off re-training than forgetting, and (2) the training process $A(\mathcal{D}, \epsilon)$ is stable, *i.e.*, if $\mathcal{D}$ and $\mathcal{D}'$ differ by a few samples, then the outcome of training $A(\mathcal{D}, \epsilon)$ is close to $A(\mathcal{D}', \epsilon)$. Under stable training, we expect the two distributions $P(\mathrm{S}(w)|\mathcal{D})$ and $P(\mathrm{S}_0(w)|\mathcal{D}_r)$ in eq. (3) to be close, making forgetting easier. Indeed, we now show how we can exploit the stability of the learning algorithm to bound the Forgetting Lagrangian.

Proposition 2 (Local Forgetting Bound)**.** Let $A(\mathcal{D}, \epsilon)$ be a training algorithm with random seed $\epsilon \sim P(\epsilon)$. Notice that in this case $P(S(w)|\mathcal{D}) = \mathbb{E}_\epsilon[P(S(w)|\mathcal{D}, \epsilon)]$. We then have the bound:

$$\mathrm{KL}\left(P(S(w)|\mathcal{D}) \,\|\, P(S_0(w)|\mathcal{D}_r)\right) \leq \mathbb{E}_\epsilon\Big[\,\mathrm{KL}\left(P(S(w)|\mathcal{D}, \epsilon) \,\|\, P(S_0(w)|\mathcal{D}_r, \epsilon)\right)\Big]$$

In the local forgetting bound we do not look at the global distribution of possible outcomes as the random seed varies, but only at the average of forgetting using a particular random seed. To see the value of this bound, consider the following example.

Corollary 1 (Gaussian forgetting)**.** Consider the case where $S(w) = h(w) + n$ and $S_0(w) = w + n'$, where $n, n' \sim \mathcal{N}(0, \Sigma)$ is Gaussian noise and $h(w)$ is a deterministic function. Since for a fixed random seed ϵ the weights $w = A(\mathcal{D}, \epsilon)$ are a deterministic function of the data, we have $P(S(w)|\mathcal{D}, \epsilon) = \mathcal{N}(h(A(\mathcal{D}, \epsilon)), \Sigma)$ and similarly $P(S_0(w)|\mathcal{D}_r, \epsilon) = \mathcal{N}(A(\mathcal{D}_r, \epsilon), \Sigma)$. Then, using the previous bound, we have:

$$\mathrm{KL}\left(P(S(w)|\mathcal{D}) \,\|\, P(S_0(w)|\mathcal{D}_r)\right) \leq \frac{1}{2}\mathbb{E}_\epsilon\Big[(h(w) - w')^T\Sigma^{-1}(h(w) - w')\Big] \quad (5)$$

where $w = A(\mathcal{D}, \epsilon)$ and $w' = A(\mathcal{D}_r, \epsilon)$.

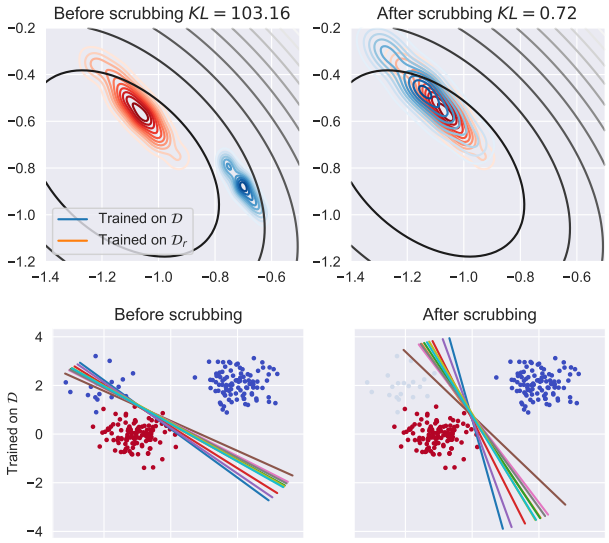

Figure 1. **(Top)** Distributions of weights $P(w|\mathcal{D})$ and $P(w|\mathcal{D}_r)$ before and after the scrubbing procedure is applied to forget the samples $\mathcal{D}_f$. The scrubbing procedure makes the two distributions indistinguishable, thus preventing an attacker from extracting any information about $\mathcal{D}_f$. The KL divergence measures the maximum amount of information that an attacker can extract. After forgetting, less than 1 NAT of information about the cohort $\mathcal{D}_f$ is accessible. **(Bottom)** The effect of the scrubbing procedure on the distribution of possible classification boundaries obtained after training. After forgetting the subject on the top left blue cluster, the classification boundaries adjust as if she never existed, and the distribution mimics the one that would have been obtained by training from scratch without that data.

That is, we can upper-bound the complex term $\mathrm{KL}\left(P(S(w)|\mathcal{D}) \,\|\, P(S_0(w)|\mathcal{D}_r)\right)$ with a much simpler one obtained by averaging the results of training and scrubbing with different random seeds.

Moreover, this suggests three simple but general procedures to forget. Under the stability assumption, we can either (i) apply a function $h(w)$ that bring w and w' closer together (*i.e.*, minimize $h(w) - w'$ in eq. (5)), or (ii) add noise whose covariance Σ is high in the direction $h(w) - w'$, or (iii) both. Indeed, this will be the basis of our forgetting algorithm, which we describe next.

3. Optimal Quadratic Scrubbing Algorithm

In this section, we derive an optimal scrubbing algorithm under a local quadratic approximation. We then validate the method empirically in complex real world problems where the assumptions are violated. We start with strong assumptions, namely that the loss is quadratic and optimized in the limit of small learning rate, giving the continuous gradient descent optimization

$$A_t(\mathcal{D}, \epsilon) = w_0 - (I - e^{-\eta A t})A^{-1}\nabla_w L_{\mathcal{D}}(w)|_{w=w_0},$$

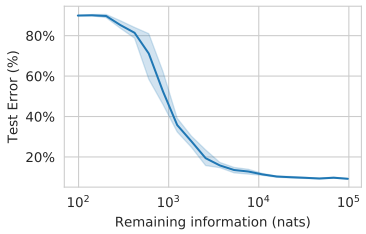

Figure 2. Trade-off between information remaining about the class to forget and test error, mediated by the parameter λ in the Lagrangian: We can always forget more, but this comes at the cost of decreased accuracy.

where $A = \nabla^2 L_{\mathcal{D}}(w)$ is the Hessian of the loss. We will relax these assumptions later.

Proposition 3 (Optimal quadratic scrubbing algorithm)**.** Let the loss be $L_{\mathcal{D}}(w) = L_{\mathcal{D}_f}(w) + L_{\mathcal{D}_r}(w)$, and assume both $L_{\mathcal{D}}(w)$ and $L_{\mathcal{D}_r}(w)$ are quadratic. Assume that the optimization algorithm $A_t(\mathcal{D}, \epsilon)$ at time t is given by the gradient flow on the loss with random initialization. Consider the scrubbing function

$$h(w) = w + e^{-Bt}e^{At}d + e^{-Bt}(d - d_r) - d_r,$$

where $A = \nabla^2 L_{\mathcal{D}}(w)$, $B = \nabla^2 L_{\mathcal{D}_r}(w)$, $d = A^{-1}\nabla_w L_{\mathcal{D}}$ and $d_r = B^{-1}\nabla_w L_{\mathcal{D}_r}$. Then, $h(w)$ is such that $h(A_t(\mathcal{D}, \epsilon)) = A_t(\mathcal{D}_r, \epsilon)$ for all random initializations ϵ and all times t. In particular, $S(w) = h(w)$ scrubs the model clean of all information in $\mathcal{D}_f$:

$$\mathrm{KL}\left(P(S(w)|\mathcal{D}, \epsilon) \,\|\, P(w|\mathcal{D}_r, \epsilon)\right) = 0.$$

Note that when $t \to \infty$, that is, after the optimization algorithm has converged, this reduces to the simple *Newton update*:

$$S_\infty(w) = w - B^{-1}\nabla L_{\mathcal{D}_r}(w).$$

3.1. Robust Scrubbing

Proposition 3 requires the loss to be quadratic, which is typically not the case. Even if it was, practical optimization proceeds in discrete steps, not as a gradient flow. To relax these assumptions, we exploit the remaining degree of freedom in the general scrubbing procedure introduced in Corollary 1, which is the noise.

Proposition 4 (Robust scrubbing procedure)**.** Assume that $h(w)$ is close to w' up to some normally distributed error $h(w) - w' \sim N(0, \Sigma_h)$, and assume that $L_{\mathcal{D}_r}(w)$ is (locally) quadratic around $h(w)$. Then the optimal scrubbing procedure in the form $S(w) = h(w) + n$, with $n \sim N(0, \Sigma)$, that minimizes the Forgetting Lagrangian eq. (4) is obtained when $\Sigma B \Sigma = \lambda \Sigma_h$, where $B = \nabla^2 L_{\mathcal{D}_r}(w)$. In particular, if the error is isotropic, that is $\Sigma_h = \sigma_h^2 I$ is a multiple of the identity, we have $\Sigma = \sqrt{\lambda \sigma_h^2} B^{-1/2}$.

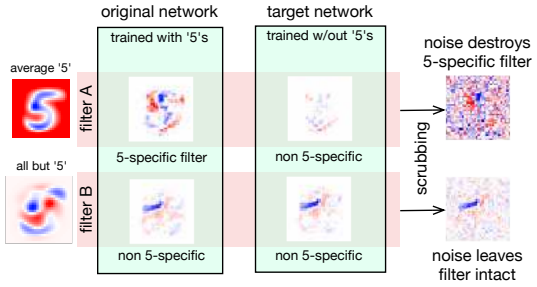

Figure 3. Filters of a network trained with the same random seed, with and without 5's. Some filters specialize to be 5-specific (filter A), and differ between the two networks, while others are not 5-specific (filter B), and remain identical. The scrubbing procedure brings original and target network closer by destroying 5-specific filters, effectively removing information about 5's.

Putting this together with the result in Proposition 3 gives us the following robust scrubbing procedure:

$$S_t(w) = w + e^{-Bt}e^{At}d + e^{-Bt}(d - d_r) - d_r + (\lambda\sigma_h^2)^{\frac{1}{4}} B^{-1/4} n, \quad (6)$$

where $n \sim N(0, I)$ and B, d and d_r are as in Proposition 3. In Figure 1 we show the effect of the scrubbing procedure on a simple logistic regression problem (which is not quadratic) trained with SGD (which does not satisfy the gradient flow assumption). Nonetheless, the scrubbing procedure manages to bring the value of the KL divergence close to zero. Finally, when $t \to \infty$ (*i.e.*, the optimization is near convergence), this simplifies to the noisy Newton update which can be more readily applied:

$$\boxed{S_t(w) = w - B^{-1}\nabla L_{\mathcal{D}_r}(w) + (\lambda\sigma_h^2)^{\frac{1}{4}} B^{-1/4}\epsilon.} \quad (7)$$

Here λ is a hyperparameter that trades off residual information about the data to be forgotten, and accuracy on the data to be retained. The hyperparameter σ_h reflect the error in approximating the SGD behavior with a continuous gradient flow.

3.2. Forgetting using a subset of the data

Once a model is trained, a request to forget $\mathcal{D}_f$ may be initiated by providing that cohort, as in the fictional service of Lacuna INC, but in general one may no longer have available the remainder of the dataset used for training, $\mathcal{D}_r$. However, assuming we are at a minimum of $L_{\mathcal{D}}(w)$, we have $\nabla L_{\mathcal{D}}(w) = 0$. Hence, we can rewrite $\nabla L_{\mathcal{D}_r}(w) = -\nabla L_{\mathcal{D}_f}(w)$ and $\nabla^2 L_{\mathcal{D}_f}(w) = \nabla^2 L_{\mathcal{D}}(w) - \nabla^2 L_{\mathcal{D}_r}(w)$. Using these identities, instead of recomputing the gradients and Hessian on the whole dataset, we can simply use those computed on the cohort to be forgotten, provided we cached the Hessian $\nabla^2 L_{\mathcal{D}}(w)$ we obtained at the end of the training on the original dataset $\mathcal{D}$. Note that this is not a requirement,

although recommended in case the data to be remembered is no longer available.

3.3. Hessian approximation and Fisher Information

In practice, the Hessian is too expensive to compute for a DNN. In general, we cannot even ensure it is positive definite. To address both issues, we use the Levenberg-Marquardt semi-positive-definite approximation:

$$\nabla^2 L_{\mathcal{D}}(w) \simeq \mathbb{E}_{x\sim\mathcal{D},y\sim p(y|x)}[\nabla_w \log p_w(y|x) \nabla_w \log p_w(y|x)^T]. \tag{8}$$

This approximation of the Hessian coincides with the Fisher Information Matrix (FIM) [20], which opens the door to information-theoretic interpretations of the scrubbing procedure. Moreover, this approximation is exact for some problems, such as linear (logistic) regression.

4. Deep Network Scrubbing

Finally, we now discuss how to robustly apply the forgetting procedure eq. (7) to deep networks. We present two variants. The first uses the FIM of the network. However, since this depends on the network gradients, it may not be robust when the loss landscape is highly irregular. To solve this, we present a more robust method that attempts to minimize directly the Forgetting Lagrangian eq. (4) through a variational optimization procedure.

Fisher forgetting: As mentioned in eq. (8), we approximate the Hessian with the Fisher Information Matrix. Since the FIM is too large to store in memory, we can compute its diagonal, or a better Kronecker-factorized approximation [21]. In our experiments, we find that the diagonal is not a good enough approximation of B for a full Newton step $h(w) = w - B^{-1}\nabla L_{\mathcal{D}_r}(w)$ in eq. (7). However, the diagonal is still a good approximation for the purpose of adding noise. Therefore, we simplify the procedure and take $h(w) = w$, while we still use the approximation of the FIM as the covariance of the noise. This results in the simplified scrubbing procedure:

$$\mathrm{S}(w) = w + (\lambda\sigma_h^2)^{\frac{1}{4}} F^{-1/4},$$

where F is the FIM (eq. 8) computed at the point w for the dataset $\mathcal{D}_r$. Here λ is a hyper-parameter that trades off forgetting with the increase in error, as shown in Figure 2. Notice that, since $h(w) = w$, instead of a Newton step, this procedure relies on w and w' already being close, which hinges on the stability of SGD. This procedure may be interpreted as adding noise to destroy the weights that may have been informative about $\mathcal{D}_f$ but not $\mathcal{D}_r$ (Figure 3).

Variational forgetting: Rather than using the FIM, we may optimize for the noise in the Forgetting Lagrangian in eq. (4): Not knowing the optimal direction $w - w'$ along which to add noise (see Corollary 1), we may add the maximum amount of noise in all directions, while keeping the increase in the loss to a minimum. Formally, we minimize the proxy Lagrangian:

$$\mathcal{L}(\Sigma) = \mathbb{E}_{n\sim N(0,\Sigma)}\left[L_{\mathcal{D}_r}(w+n)\right] - \lambda \log|\Sigma|.$$

The optimal Σ may be seen as the FIM computed over a smoothed landscape. Since the noise is Gaussian, $\mathcal{L}(\Sigma)$ can be optimized using the local reparametrization trick [16].

5. Experiments

We report experiments on MNIST, CIFAR10 [18], Lacuna-10 and Lacuna-100, which we introduce and consist respectively of faces of 10 and 100 different celebrities from VGGFaces2 [6] (see Appendix for details). On both CIFAR-10 and Lacuna-10 we choose to forget either an entire class, or a hundred images of the class.

For images (Lacuna-10 and CIFAR10), we use a small All-CNN (reducing the number of layers) [27], to which we add batch normalization before each non-linearity. We pre-train on Lacuna-100/CIFAR-100 for 15 epochs using SGD with fixed learning rate of 0.1, momentum 0.9 and weight decay 0.0005. We fine-tune on Lacuna-10/CIFAR-10 with learning rate 0.01. To simplify the analysis, during fine-tuning we do not update the running mean and variance of batch normalization, and rather reuse the pre-trained ones.

5.1. Linear logistic regression

First, to validate the theory, we test the scrubbing procedure in eq. (6) on logistic regression, where the task is to forget data points belonging to one of two clusters comprising the class (see Figure 1). We train using a uniform random initialization for the weights and SGD with batch size 10, with early stopping after 10 epochs. Since the problem is low-dimensional, we easily approximate the distribution $p(w|\mathcal{D})$ and $p(w|\mathcal{D}_r)$ by training 100 times with different random seeds. As can be seen in Figure 1, the scrubbing procedure is able to align the two distributions with near perfect overlap, therefore preventing an attacker form extracting any information about the forgotten cluster. Notice also that, since we use early stopping, the algorithm had not yet converged, and exploiting the time dependency in eq. (6) rather than using the simpler eq. (7) is critical.

5.2. Baseline forgetting methods

Together with our proposed methods, we experiment with four other baselines which may intuitively provide some degree of forgetting. (i) **Fine-tune**: we fine-tune the model on the remaining data $\mathcal{D}_r$ using a slightly large learning rate. This is akin to catastrophic forgetting, where fine-tuning without $\mathcal{D}_f$ may make the model forget the original solution to $\mathcal{D}_f$ (more so because of the larger learning rate). (ii) **Negative Gradient**: we fine-tune on $\mathcal{D}$ by

	Metrics	**Original model**	**Retrain** (target)	**Finetune**	**Neg. Grad.**	**Rand. Lbls.**	**Hiding**	**Fisher** (ours)	**Variational** (ours)
Lacuna-10	Error on $\mathcal{D}_{\text{test}}$ (%)	10.2 ±0.5	10.3 ±0.4	10.2 ±0.6	10.0 ±0.4	12.0 ±0.2	18.2 ±0.4	14.5 ±1.6	13.7 ±1.0
Scrub 100	Error on $\mathcal{D}_f$ (%)	0.0 ±0.0	15.3 ±0.6	0.0 ±0.0	0.0 ±0.0	6.0 ±3.6	100 ±0.0	8.0 ±2.7	8.0 ±3.6
All-CNN	Error on $\mathcal{D}_r$ (%)	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.1 ±0.1	6.5 ±0.0	4.8 ±2.8	4.8 ±2.4
	Info-bound (kNATs)							3.3 ±1.1	3.0 ±0.5
Lacuna-10	Error on $\mathcal{D}_{\text{test}}$ (%)	10.2 ±0.5	18.4 ±0.6	10.0 ±0.6	18.4 ±0.6	18.8 ±0.6	18.2 ±0.4	21.0 ±1.3	20.9 ±0.4
Forget class	Error on $\mathcal{D}_f$ (%)	0.0 ±0.0	100 ±0.0	0.0 ±0.0	100 ±0.2	90.2 ±1.5	100.0 ±0.0	100.0 ±0.0	100.0 ±0.0
All-CNN	Error on $\mathcal{D}_r$ (%)	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	3.3 ±2.3	2.8 ±1.4
	Info-bound (kNATs)							13.2 ±2.8	12.0 ±2.9
CIFAR-10	Error on $\mathcal{D}_{\text{test}}$ (%)	14.4 ±0.6	14.6 ±0.7	13.5 ±0.1	13.4 ±0.1	13.8 ±0.1	21.0 ±0.5	19.8 ±2.8	20.9 ±4.8
Scrub 100	Error on $\mathcal{D}_f$ (%)	0.0 ±0.0	19.3 ±4.5	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	100.0 ±0.0	23.3 ±2.1	6.3 ±2.5
All-CNN	Error on $\mathcal{D}_r$ (%)	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	9.9 ±0.1	8.0 ±4.3	8.8 ±5.2
	Info-bound (kNATs)							33.4 ±16.7	21.6 ±5.2
CIFAR-10	Error on $\mathcal{D}_{\text{test}}$ (%)	14.4 ±0.7	21.1 ±0.6	14.3 ±0.1	20.2 ±0.1	20.7 ±0.4	21.0 ±0.5	23.7 ±0.9	22.8 ±0.3
Forget class	Error on $\mathcal{D}_f$ (%)	0.0 ±0.0	100 ±0.0	10.0 ±0.4	100 ±0.2	88.1 ±4.6	100.0 ±0.0	100.0 ±0.0	100.0 ±0.0
All-CNN	Error on $\mathcal{D}_r$ (%)	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	2.6 ±1.8	2.3 ±0.7
	Info-bound (kNATs)							458.1 ±172.2	371.5 ±51.3

Table 1. **Original model** is the model trained on all data $\mathcal{D} = \mathcal{D}_f \sqcup \mathcal{D}_r$. The forgetting algorithm should scrub information from its weights. **Retrain** denotes the model obtained by retraining from scratch on $\mathcal{D}_r$, without knowledge of $\mathcal{D}_f$. The metric values in the Retrain column is the optimal value which every other scrubbing procedure should attempt to match. We consider the following forgetting procedures: **Fine-tune** denotes fine-tuning the model on $\mathcal{D}_r$. **Negative Gradient** (Neg. Grad.) denotes fine-tuning on $\mathcal{D}_f$ by moving in the direction of increasing loss. **Random Label** (Rnd. Lbls.) denotes replacing the labels of the class with random labels and then fine-tuning on all $\mathcal{D}$. **Hiding** denotes simply removing the class from the final classification layer. **Fisher** and **Variational** are our proposed methods, which add noise to the weights to destroy information about $\mathcal{D}_f$ following the Forgetting Lagrangian. We benchmark these methods using several readout functions: errors on $\mathcal{D}_f$ and $\mathcal{D}_r$ after scrubbing, time to retrain on the forgotten samples after scrubbing, distribution of the model entropy. In all cases, the read-out of the scrubbed model should be closer to the target retrained model than to the original. Note that our methods also provide an upper-bound to the amount of information remaining. We report mean/std over 3 random seeds.

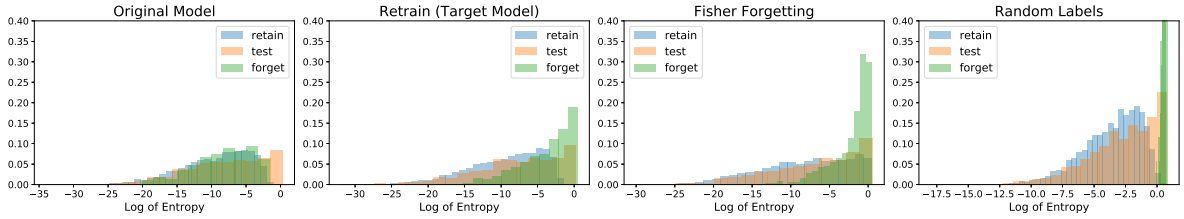

Figure 4. **Streisand Effect:** Distribution of the entropy of model output (confidence) on: the retain set $\mathcal{D}_r$, the forget set $\mathcal{D}_f$, and the test set. The **original model** has seen $\mathcal{D}_f$, and its prediction on it are very confident (matching the confidence on the train data). On the other hand, a model **re-trained** without seeing $\mathcal{D}_f$ has a lower confidence $\mathcal{D}_f$. After applying our scrubbing procedures (**Fisher** and **Variational**) to the original model, the confidence matches more closely the one we would have expected for a model that has never seen the data (column 3 is more similar to 2 than 1). For an incorrect method of forgetting, like training with random labels, we observe that the entropy of the forgotten samples is very degenerate and different from what we would have expected if the model had actually never seen those samples (it is concentrated only around chance level prediction entropy). That is, attempting to remove information about a particular cohort using this method, may actually end up providing more information about the cohort than the original model.

moving in the direction of increasing loss for samples in $\mathcal{D}_f$, which is equivalent to using a negative gradient for the samples to forget. This aims to damage features predicting $\mathcal{D}_f$ correctly. (iii) **Random Labels**: fine-tune the model on $\mathcal{D}$ by randomly resampling labels corresponding to images belonging to $\mathcal{D}_f$, so that those samples will get a random gradient. (iv) **Hiding**: we simply remove the row corresponding to the class to forget from the final classification layer of the DNN.

5.3. Readout functions used

Unlike our methods, these baselines do not come with an upper bound on the quantity of remaining information. It is therefore unclear how much information is removed. For this reason, we introduce the following *read-out functions*, which may be used to gauge how much information they were able destroy: (i) **Error on the test set** $\mathcal{D}_{\text{test}}$ (ideally small), (ii) **Error on the cohort to be forgotten** $\mathcal{D}_f$ (ideally the same as a model trained without seeing $\mathcal{D}_f$), (iii) **Error on the residual** $\mathcal{D}_r$ (ideally small), (iv) **Re-learn**

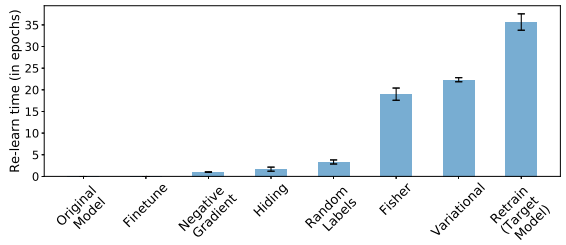

Figure 5. **Re-learn time** (in epochs) for various forgetting methods. All the baselines method can quickly recover perfect performance on $\mathcal{D}_f$, suggesting that they do not actually scrub information from the weights. On the other hand, the relearn time for our methods is higher, and closer to the one of a model that has never seen the data, suggesting that they remove more information.

time (in epochs) time to retrain the scrubbed model on the forgotten data (measured by the time for the loss to reach a fixed threshold, ideally slow). If a scrubbed model can quickly recover a good accuracy, information about that cohort is likely still present in the weights. (v) **Model confidence:** We plot the distribution of model confidence (entropy of the output prediction) on the retain set $\mathcal{D}_r$, forget set $\mathcal{D}_f$ and the test set (should look similar to the confidence of a model that has never seen the data). (vi) **Information bound**: For our methods, we compute the information upper-bound about the cohort to be forgotten in NATs using Proposition 2.

5.4. Results

First, in Table 1 we show the results of scrubbing $\mathcal{D}_f$ from model trained on all the data. We test both the case we want to forget only a subset of 100-images from the class, and when we want to forget a whole identity. We test on CIFAR-10 and Lacuna-10 with a network pretrained on CIFAR-100 and Lacuna-100 respectively.

Retrain denotes the gold standard which every scrubbing procedure should attempt to match for the error readout function. From the case where we want to scrub a subset of a class (first and third row of Retrain) it is clear that scrubbing does not mean merely achieving 100% error on $\mathcal{D}_f$. In fact, the reference Retrain has 15.3% and 19.3% error respectively on $\mathcal{D}_f$ and not 100%. Rather it means removing the information from the weights so that it behaves identically to a re-trained model. Forgetting by fine-tuning on $\mathcal{D}_r$, performs poorly on the error readout function (error on $\mathcal{D}_f$ and $\mathcal{D}_r$), suggesting that using catastrophic forgetting is the not the correct solution to selective forgetting.

The Negative Gradient and Random Labels methods perform well on the error readout function, however, when we use the re-learn time as a readout function (Figure 5) it becomes clear that very little information is actually removed, as the model relearn $\mathcal{D}_f$ very quickly. This suggests that merely scrubbing the activations by hiding or changing some output is not sufficient for selective forgetting; rather, information needs to be removed from the weights as anticipated. Moreover, applying an incorrect scrubbing procedure may make the images to forget *more* noticeable to an attacker (Streisand effect), as we can see by from the confidence values in Figure 4. The ease of forgetting a learnt cohort also depends on its size. In particular, in Figure 6 we observe that, for a fixed value of λ in eq. (1), the upper-bound on the information retained by the model after scrubbing increases with the size of the cohort to forget.

6. Discussion

Our approach is rooted in the connection between Differential Privacy (which our framework generalizes) and the stability of SGD. Forgetting is also intrinsically connected with information: Forgetting may also be seen as minimizing an upper-bound on the amount of information that the weights contain about $\mathcal{D}_f$ [3] and that an attacker may extract about that particular cohort $\mathcal{D}_f$ using some readout function f. We have studied this problem from the point of view of Shannon Information, which allows for an easy formalization. However, it also has the drawback of considering the worst case of an attacker that has full knowledge of the training procedure and can use arbitrarily complex readout functions which may, for example, simulate all possible trainings of the network to extract the result. Characterizing forgetting with respect to a viable subset of realistic readout functions $f(w)$ is a promising area of research. We also exploit stability of the training algorithm after pretraining. Forgetting without the pretraining assumption is an interesting challenge, as it has been observed that slight perturbation of the initial critical learning period can lead to large difference in the final solution [2, 12].

Acknowledgements: We would like to thank the anonymous reviewers for their feedback and suggestions. This work is supported by ARO W911NF-17-1-0304, ONR N00014-17-1-2072, ONR N00014-19-1-2229, ONR N00014-19-1-2066.

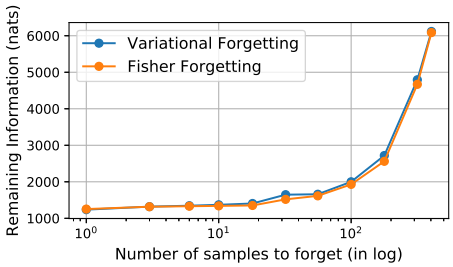

Figure 6. **Difficulty of forgetting increases with cohort size.** For a fixed λ (forgetting parameter), we plot the amount of information remaining after scrubbing as a function of the cohort size ($|\mathcal{D}_f|$).

References

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318. ACM, 2016. 1

[2] Alessandro Achille, Matteo Rovere, and Stefano Soatto. Critical learning periods in deep neural networks. In *International Conference of Learning Representations*, 2019. 8

[3] Alessandro Achille and Stefano Soatto. Where is the Information in a Deep Neural Network? *arXiv e-prints*, page arXiv:1905.12213, May 2019. 2, 8

[4] Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 233–242. JMLR. org, 2017. 1

[5] Lucas Bourtoule, Varun Chandrasekaran, Christopher Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. *arXiv preprint arXiv:1912.03817*, 2019. 2

[6] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman. Vggface2: A dataset for recognising faces across pose and age. In *International Conference on Automatic Face and Gesture Recognition*, 2018. 6

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

[8] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014. 1

[9] Gintare Karolina Dziugaite and Daniel M Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *arXiv preprint arXiv:1703.11008*, 2017. 2

[10] 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*, pages 1322–1333. ACM, 2015. 2

[11] Antonio Ginart, Melody Guan, Gregory Valiant, and James Y Zou. Making ai forget you: Data deletion in machine learning. In *Advances in Neural Information Processing Systems*, pages 3513–3526, 2019. 2

[12] Aditya Sharad Golatkar, Alessandro Achille, and Stefano Soatto. Time matters in regularizing deep networks: Weight decay and data augmentation affect early learning dynamics, matter little near convergence. In *Advances in Neural Information Processing Systems*, pages 10677–10687, 2019. 8

[13] Moritz Hardt, Benjamin Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. *arXiv preprint arXiv:1509.01240*, 2015. 2

[14] Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. Logan: Membership inference attacks against generative models. *Proceedings on Privacy Enhancing Technologies*, 2019(1):133–152, 2019. 1

[15] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. Deep models under the gan: information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 603–618. ACM, 2017. 1

[16] Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, pages 2575–2583, 2015. 6

[17] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017. 2

[18] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 6

[19] Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in neural information processing systems*, pages 598–605, 1990. 2

[20] James Martens. New insights and perspectives on the natural gradient method. *arXiv preprint arXiv:1412.1193*, 2014. 6

[21] James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pages 2408–2417, 2015. 6

[22] David McAllester. A pac-bayesian tutorial with a d ropout bound. *arXiv preprint arXiv:1307.2118*, 2013. 2

[23] Paul Micaelli and Amos Storkey. Zero-shot knowledge transfer via adversarial belief matching. *arXiv preprint arXiv:1905.09768*, 2019. 2

[24] Apostolos Pyrgelis, Carmela Troncoso, and Emiliano De Cristofaro. Knock knock, who's there? membership inference on aggregate location data. *arXiv preprint arXiv:1708.06145*, 2017. 1

[25] Saurabh Shintre and Jasjeet Dhaliwal. Verifying that the influence of a user data point has been removed from a machine learning classifier, Mar. 2019. US Patent App. 10/225,277. 2

[26] Congzheng Song, Thomas Ristenpart, and Vitaly Shmatikov. Machine learning models that remember too much. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 587–601. ACM, 2017. 1

[27] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014. 6

[28] Stacey Truex, Ling Liu, Mehmet Emre Gursoy, Lei Yu, and Wenqi Wei. Demystifying membership inference attacks in machine learning as a service. *IEEE Transactions on Services Computing*, 2019. 1

[29] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016. 1