Deep Unlearning via Randomized Conditionally Independent Hessians

Ronak Mehta*[1] ronakrm@cs.wisc.edu
Sourav Pal*[1] spal9@wisc.edu
Vikas Singh[1] vsingh@biostat.wisc.edu
Sathya N. Ravi[2] sathya@uic.edu

[1]University of Wisconsin-Madison [2]University of Illinois at Chicago

Abstract

Recent legislation has led to interest in machine unlearning*, i.e., removing specific training samples from a* predictive *model as if they never existed in the training dataset. Unlearning may also be required due to corrupted/adversarial data or simply a user's updated privacy requirement. For models which require no training (k-NN), simply deleting the closest original sample can be effective. But this idea is inapplicable to models which learn richer representations. Recent ideas leveraging optimization-based updates scale poorly with the model dimension d, due to inverting the Hessian of the loss function. We use a variant of a new conditional independence coefficient, L-CODEC, to identify a subset of the model parameters with the most semantic overlap on an individual sample level. Our approach completely avoids the need to invert a (possibly) huge matrix. By utilizing a Markov blanket selection, we premise that L-CODEC is also suitable for deep unlearning, as well as other applications in vision. Compared to alternatives, L-CODEC makes approximate unlearning possible in settings that would otherwise be infeasible, including vision models used for face recognition, person re-identification and NLP models that may require unlearning samples identified for exclusion. Code is available at* https://github.com/vsingh-group/LCODEC-deep-unlearning

1. Introduction

As personal data becomes a valuable commodity, legislative efforts have begun to push back on its widespread collection/use particularly for training ML models. Recently, a focus is the "right to be forgotten" (RTBF), i.e., the right of an individual's data to be deleted from a database (and derived products). Despite existing legal frameworks on fair use, industry scraping has led to personal images being used without consent, e.g. [20]. Large datasets are not only stored for descriptive statistics, but used in training large models. While regulation (GDPR, CCPA) has not specified the extent to which data must be forgotten, it poses a clear question: is deletion of the data enough, or does a model trained on that data also needs to be updated?

Recent work by [6, 7] has identified scenarios where trained models are vulnerable to attacks that can reconstruct input training data. More directly, recent rulings by the Federal Trade Commission [12, 24] have ordered companies to fully delete and destroy not only data, but also any model trained using those data. While deletion and (subsequent) full model retraining without the deleted samples is possible, most in-production models require weeks of training and review, with extensive computational/human resource cost. With additional deletions, it is infeasible to retrain each time a new delete request comes in. So, how to update a model ensuring the data is deleted without retraining?

Task. Given a set of input data $\mathcal{S} : \{z_i\}_{i=1}^n \sim \mathcal{D}$ of size n, training simply identifies a hypothesis $\hat{w} \in \mathcal{W}$ via an iterative scheme $w_{t+1} = w_t - g(\hat{w}, z')$ until convergence, where $g(\cdot, z')$ is a stochastic gradient of a fixed loss function. Once a model at convergence is found, *machine unlearning* aims to identify an update to $\hat{w}$ through an analogous *one-shot unlearning update*:

$$w' = \hat{w} + g_{\hat{w}}\left(z'\right), \tag{1}$$

for a *given* sample $z' \in \mathcal{S}$ that is to be **unlearned**.

Contributions. We address several computational issues with existing approximate formulations for unlearning by taking advantage of a new statistical scheme for sufficient parameter selection. First, in order to ensure that a sample's impact on the model predictions is minimized, we propose a measure for computing conditional independence called L-CODEC which identifies the Markov Blanket of parameters to be updated. Second, we show that the L-CODEC identified Markov Blanket enables unlearning in previously infeasible deep models, scaling to networks with hundreds of millions of parameters. Finally, we demonstrate the ability of L-CODEC to unlearn samples and entire classes on networks, from CNNs/ResNets to transformers, including face recognition and person re-identification models.

*Joint First Authors.

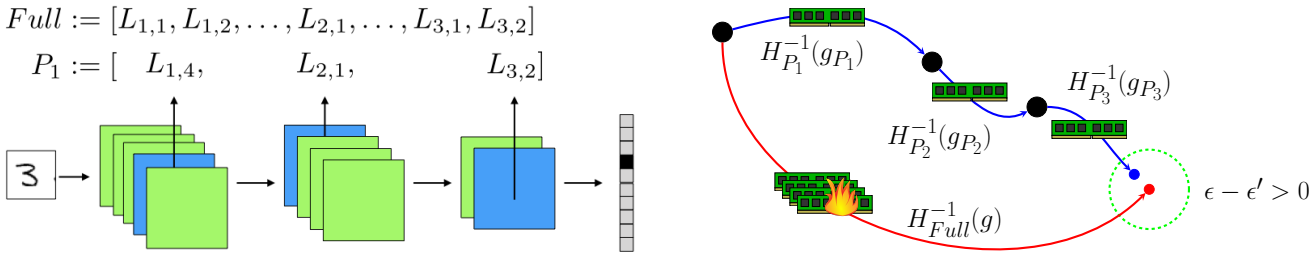


Figure 1. Large deep learning networks typically associate specific subsets of network parameters, blocks (blue), to specific samples in the input space. Traditional forward or backward passes may not reveal these blocks: high correlations among features may not distinguish important ones. Input perturbations can be used to identify them in a probabilistic, distribution-free manner. These blocks can then be unlearned together in an efficient block-coordinate style update (right, blue lines), approximating an update to the full network which requires a costly/infeasible full Hessian inverse (red line).

2. Problem Setup for Unlearning

Let $\mathcal{A}$ be an algorithm that takes as input a training set $\mathcal{S}$ and outputs a hypothesis $w \in \mathcal{W}$, defined by a set of d parameters Θ. An unlearning scheme $\mathcal{U}$ takes as input a sample $z' \in \mathcal{S}$ used as input to $\mathcal{A}$, and ideally, outputs an **updated** hypothesis $w' \in \mathcal{W}$ where z' has been deleted from the model. An unlearning algorithm should output a hypothesis that is close or equivalent to one that would have been learned had the input to $\mathcal{A}$ been $\mathcal{S} \setminus z'$. A framework for this goal was given by [13] as,

Definition 1 ($(\epsilon, \delta)-$ forgetting)**.** *For all sets $\mathcal{S}$ of size n, with a "delete request" $z' \in \mathcal{S}$, an unlearning algorithm $\mathcal{U}$ is $(\epsilon, \delta)-$forgetting if*

$$\mathbb{P}(\mathcal{U}(\mathcal{A}(S), z') \in \mathcal{W}) \leq e^{\epsilon}\mathbb{P}(\mathcal{A}(\mathcal{S} \setminus z') \in \mathcal{W}) + \delta \quad (2)$$

In essence, for an existing model w, a good unlearning algorithm for request $z' \in \mathcal{S}$ will output a model $\hat{w}$ close to the output of $\mathcal{A}(S \setminus z')$ with high probability.

Remark 1. *Definition 1 is similar to the standard definitions of differential privacy. The connection to unlearning is: if an algorithm is $(\epsilon, \delta)-$forgetting for unlearning, then it is also differentially private.*

If $\mathcal{A}$ is an empirical risk minimizer for the loss f, let

$$\mathcal{A} : (\mathcal{S}, f) \to \hat{w} \quad (3)$$

$\hat{w} = \arg\min F(w)$ and $F(w) = \frac{1}{n}\sum_{i=1}^{n} f(w, z_i)$. Recall $g(z')$ from (1): our unlearning task essentially involves identifying the form of $g(z')$ for which the update in (1) is (ϵ, δ)-forgetting. If an oracle provides this information, we have accomplished the unlearning task.

The difficulty, as expected, tends to depend on f and $\mathcal{A}$. Recent unlearning results have identified forms of f and $\mathcal{A}$ where such a $g(z')$ exists. The authors in [30] define $g(z') = \frac{1}{n-1}H'^{-1}\nabla f(\hat{w}, z')$, where

$$H' = \frac{1}{n-1}\left(n\nabla^2 F(\hat{w}) - \nabla^2 f(\hat{w}, z')\right), \quad (4)$$

with additive Gaussian noise $w' = w' + N(0, \sigma^2)$ scaling as a function of n, ϵ, δ, and the Lipschitz and (strong) convexity parameters of the loss f. We can interpret the update using (4) from the optimization perspective as a trajectory "reversal": starting at a random initialization, the first order (stochastic gradient) trajectory of w (possibly) *with* z' is reversed using *residual* second order curvature information (Hessian) at the optimal $\hat{w}$ in (4), achieving unlearning. This is shown to satisfy Def. 1, and only incurs an additive error that scales by $O(\sqrt{d}/n^2)$ in the gap between $F(w')$ and the global minimizer $F(w^*)$ over the ERM $F(\hat{w})$.

Rationale for approximate schemes. From the reversal of w optimization perspective, it is clear that there may be other choices to achieve unlearning. For a practitioner interested in unlearning, the aforementioned algorithm (as in (4)) can be directly instantiated if one has extensive computational resources. Indeed, in settings where it is not directly possible to compute the Hessian inverse necessary for $H'^{-1}\nabla f(\hat{w}, z')$, we must consider alternatives.

A potential idea. Our goal is to identify a form of $g(z')$ that **approximates** $H'^{-1}\nabla f(\hat{w}, z')$. Let us consider the Newton-style update suggested by (4) as a smoothing of a traditional first order gradient step. The inverse Hessian is a weighting matrix, appropriately scaling the gradients based on the second order difference between the training set mean point $F(\hat{w})$ and at the sample of interest $f(\hat{w}, z')$. This smoothing can also be seen from an information perspective: the Hessian in this case corresponds to a Fisher-style information matrix, and its inverse as a conditional covariance matrix [14, 16]. It is not hard to imagine that from this perspective, if there are *specific set of parameters* that have *small gradients* at $f(\hat{w}, z')$ or if the information matrix is zero or small, then we need not consider their effect.

Examples of this intuition in vision. [3, 11, 32] and others have shown that models trained on complex tasks tend to *delegate* subnetworks to specific regions of the input space. That is, parameters and functions within networks tend to (or can be encouraged to) act in *blocks*. For example, activa-

tion maps for different filters in a trained (converged) CNN model show differences for different classes, especially for filters closer to the output layer. We formalize this observation as an assumption for samples in the training set.

Assumption 1. *For all subsets of training samples $S \subset \mathcal{S}$, there exists a subset of trained model parameters $P^* \subset \Theta$ such that*

$$f(S) \perp w^*_{\Theta \setminus P} | w^*_P \tag{5}$$

Due to the computational issues discussed above, if we could make such a simple/principled selection scheme practical, it may offer significant benefits.

3. Related Work

To contextualize our contributions, we briefly review existing proposals for machine unlearning.

Naïve, Exact Unlearning. A number of authors have proposed methods for exact unlearning, in the case where $(\epsilon = 0, \delta = 0)$. SVMs by [23, 28], Naïve Bayes Classifiers by [5], and k-means methods by [13] have all been studied. But these algorithms do not translate to stochastic models with millions of parameters.

Approximate Unlearning. With links to fields such as robustness and privacy, we see more developments in approximate unlearning under Definition 1. The so-called ϵ-certified removal by [19] puts forth similar procedures when $\delta = 0$, and the model has been trained in a specific manner. [19, 22] provide updates to linear models and the last layers of networks, and [15, 16] provide updates based on linearizations that work over the full network, and follow-up work by [14] presents a scheme to unlearn under an assumption that some samples will not need to be removed.

Other recent work has taken alternative views of unlearning, which do not require/operate under probabilistic frameworks, see [4, 25]. These schemes present good guarantees in the absolute privacy setting, but they require more changes to pipelines (sharding/aggregating weaker models) and scale unsatisfactorily in large deep learning settings.

4. Randomized Markovian Block Coordinate Unlearning

If there exist entries of the vector $g(z') = H'^{-1} \nabla f(\hat{w}, z')$ that we can, through *some* procedure, identify as zero, then we can simply avoid computing such zero coordinates. Not only can we zero out those particular entries in the inverse and the gradient, but we can take advantage of the blockwise inverse to *completely remove those parameters from all computations.* If possible, it would immediately change the complexity from $O(d^3)$ to $O(p^3)$, where $p \ll d$ is the size of the subset of parameters that we know are *sufficient* to update.

Let $P \subseteq \Theta := \{1, \ldots, d\}$ be the index set of the parameters that are "sufficient" to update. A direct procedure may be to identify this subset P with

$$P = \underset{P \in \mathcal{P}(\Theta)}{\arg\min} \, ||\tilde{w} - \tilde{w}_P||, \tag{6}$$

where $\mathcal{P}(\Theta)$ is the *power set* of the elements in Θ and $\tilde{w}_P$ is the subset of the parameters we are interested in updating. Note that a simple solution to this problem *does* exist: choosing the $p = |P|$ parameters with the largest change will minimize this distance for typical norms. This can be achieved by thresholding the updates $g(z')$ for $\hat{w}$. However, this *requires computing the full update for* $g(z')$. We want a preprocessing procedure that performs the selection *before* computation of $g(z')$ is needed.

A probabilistic angle for selection. We interpret a deep network $\mathcal{W}$ as a functional on the input space $\mathcal{D}$. This perspective is common in statistics for variable selection (e.g., LASSO), albeit used *after* the entire optimization procedure is performed i.e., at the optimal solution. The only difference here is that we use it at approximately optimal solutions as given by ERM minimization. Importantly, this view allows us to identify regions in $\mathcal{W}$ that contain the most information about a query sample z'. We will formalize this intuition using recent results for conditional independence (CI) testing. Finding w_P above should also satisfy

$$z' \perp w_{\Theta \setminus P} | w_P \tag{7}$$

This CI formulation is well studied within graphical models. Many measures and hypothesis tests have been proposed to evaluate it. The *coefficient of conditional dependence* (CODEC) in [1], along with their algorithm for "feature ordering", FOCI, at first seems to offer a solution to (7), and in fact, can be implemented "as is" for shallow networks. (Review of other CI tests are in the appendix.)

Using CODEC directly for Deep Unlearning is inefficient. There are two issues: First, when applying CODEC to problems with a very large n with discrete values, the cost of tie-breaking for computing nearest neighbors can become prohibitive. Second, z' is not a random variable for which we have a number of instances. We defer discussion of the second issue to Section 5, and address the first issue here.

Consider the case where a large number of elements have an equal value. With an efficient implementation using $kd-$trees, identifying the nearest-neighbor as required by CODEC would still require expanding the nodes of all elements with equal value. As an example, if we are looking for the nearest neighbor to a point at the origin and there are a large number of elements on the surface of a sphere centered at the origin, we still require checking all entries and expanding their nodes in the tree, even when we know that they are all equal for this purpose.

Figure 2. A sample is perturbed and passed through the network. Activations are aggregated alongside losses and fed to L-FOCI. Selected rows represent slices of the corresponding layer that are sufficient for unlearning.

Interestingly, this problem has a relatively elegant solution. We introduce a randomized version of CODEC, L-CODEC. For variables A, B, C:

$$T_L := T\left(\tilde{B}, \tilde{C} | \tilde{A}\right), \tag{8}$$

where $\tilde{B} = B + N(0, \sigma^2)$, and similarly for $\tilde{C}, \tilde{A}$. This additive noise can simply be scaled to the inverse of the largest distance between any points in the set. By requiring this noise to be smaller than any distance between items in the set, the ranking will remain the same between unique discrete values, and will be perturbed slightly for equal ones. In expectation, this will still lead to the true dependence measure. The noise addition is consistent with the Randomization criterion for conditional independence – for random variables A, B, C in Borel spaces, $A \perp B | C$ *iff* $A \overset{\text{a.s.}}{=} h(B, U)$ for some measurable function h and uniform random variable $U \sim \text{Uniform}(0, 1)$ which is independent of (B, C) as in [26].

Remark 2. *An altered version of this setup also gives us a form of explainability, where we can apply sensitivity analysis to each input feature or pixel and estimate its effect on the output via a similarly randomized version of the Chatterjee rank coefficient $T(A, B)$, proposed by [8].*

4.1. Efficient Subset Selection that is also Sufficient for Predictive Purposes

The above test is good for (7) if we know *which* subset $P \in \mathcal{P}(\Theta)$ to test. Recent work by [36] proposes a selection procedure using an iterative scheme to slowly build the sufficient set, adding elements which maximally increase the information explained in the outcome of interest. While it is efficient (polynomial in size), we must know the maximal degree. A priori, we may have no knowledge of what this size is, and for parameter subsets it may be very high.

When using L-CODEC, we can use a more straightforward Markov Blanket identification procedure adapted from [1]. FOCI more directly selects which variables are valuable for explaining z', and in fact, is proven to identify the sufficient set (Markov Blanket) with a reasonable number of samples. Briefly, in our L-FOCI, the sufficient set is built incrementally with successive calls to L-CODEC, moving the most "dependent" feature from the independent set to the sufficient set. See appendix for details.

Summary. This procedure alleviates the first issue in terms of sufficient subset or Markov Blanket selection; compared to existing methods using information-theoretic measures that require permutation testing, L-FOCI directly estimates the change in variance when considering a proposal to add to the set. Now, we discuss how this selection can help identify sets of parameters that can be updated.

5. Deep Unlearning via L-FOCI Hessians

Our input samples to scrub z' are not random variables for which we have samples or distributional assumptions, nor are our parameters. In this case, a perturbation-based scheme may be useful when attempting to generate samples for unknown distributions.

Considering Assump. 1, when only some parameters are useful for the final outcome on an input sample $z' \in S$, the effect of those parameters can be measured through activations due to the forward pass of a model. We estimate the conditional independence test in (5) through activations as

$$f(z') \perp a^*_{\Theta \setminus P} | a^*_P, \tag{9}$$

where a_P for some parameter subset $P \subseteq \Theta$ is defined as the linear activations generated by the forward pass through the model. This formulation relates to a generalized version of the solution in §3 of [36], where conditional mutual information is estimated via feature mappings.

As an example, if a network has linear layers $\mathcal{L}$, a simple linear layer $l \in \mathcal{L}$ with parameters $w_l \in \mathbb{R}^{a \times b}$ would have activations $a_l \in \mathbb{R}^b$, with $a_l = w_l a_{l-1}$. For each entry $a_{l,j}$ in the vector a_l, the associated parameters in the layer are $w_l[:, j]$. Thus, we break up the network into influential *slices*. These slices can be seen as a finer view of the parameter space compared to typical layerwise selection, but coarser than a fully discrete one. Next, $\mathcal{L}$ now refers to the collection of these slices, with a specific slice as l.

The tuple of variables we need samples from is now

$$\{a_1, \dots a_{|\mathcal{L}|}, \mathcal{L}(z')\} \tag{10}$$

We can obtain samples from this set by perturbing the input and consecutively collecting activations along all weight slices during the computation of the loss. For a particular perturbation $\xi^j \sim N(0, \sigma^2)$,

$$x_i^j = x_i + \xi^j; \quad l^j, a_L^j = \{l(x_i^j), a_1^j, \dots a_{|\mathcal{L}|}^j\} \tag{11}$$

The tuples (l^j, a_L^j) serve as samples for our conditional independence test,

$$(P \subseteq \Theta) = \text{L-FOCI}((l^j, a_{\mathcal{L}}^j)_{j=1}^m) \tag{12}$$

for $J := \{j \in 1, \ldots, m\}$ perturbations (see Figure 2).

In Alg. 1, the activations are collected using hooks within the forward pass. First, gradients at the last and penultimate epoch for full training are stored during the original training pass. Given a sample to unlearn, we compute L-FOCI over the perturbed activations and losses generated by the forward pass, and identify which parameter sets will be updated. We compute the approximate Hessian over these parameters via finite differences for both the full model and for the model only over the sample of interest. Finally, we apply the blockwise Newton update to the subset of parameters as in (1) with appropriate DP noise as in [30].

Algorithm 1: Unlearning via Conditional Dependence Block Selection

Data: A trained model $\hat{w}$, gradient vectors $\nabla_1 F(\hat{w}), \nabla_2 F(\hat{w})$, sample $z' \in \mathcal{S}$ to unlearn.
Result: model w' with z' removed.
1. **for** $j \in \{1, \ldots, m\}$ *perturbations* **do**
   $\xi^j \sim N(0, \sigma^2)$
   $z'^{,j} = z' + \xi^j$
   $l^j, a^j = f(z'^{,j})$
end
2. Compute $P* =$ L-FOCI(l^J, a^J).
3. Compute $\nabla^2_P F(\hat{w}, z')$ via finite differences.
4. Update:

$$H'_P = \frac{1}{n-1}\left(n\nabla^2_P F(\hat{w}) - \nabla^2_P f(\hat{w}, z')\right) \quad (13)$$

$$w'_P = \hat{w}_P + \frac{1}{n-1} H'^{-1}_P \nabla f(\hat{w}, z')_P \quad (14)$$

$$w'_{\Theta\setminus P} = \hat{w}_{\Theta\setminus P} \quad (15)$$

Computational Gains. A direct observation is that now we are doing sampling, which adds a linear computational load. However, directly updating all parameters requires $O(d^3)$ computation due to matrix inversion, while this procedure requires $O(md + dm \log m + p^3)$, for the forward passes, FOCI algorithm, and subsequent subsetted matrix inversion. For any reasonable setting, we have $p \ll d$, and so this clearly offers significant practical advantages.

5.1. Theoretical Analysis

By definition, any neural network as described above is actually a Markov Chain: we know that the output of a layer is conditionally independent of the penultimate one given the previous one, and clearly a change in one layer will propagate forward through the rest of the network. However, when trained for a task with a large number of samples, the influence or "memory" of the network with respect to a specific sample may not be clear. While the output of the layers may follow a Markov Chain, the parameters in the layers themselves do not, and their influence on a sample through the forward pass may be highly dependent or correlated. Practically, we would hope that unlearning samples at convergence does not cause too much damage to the model's performance on the rest of the input samples. Following traditional unlearning analysis, we can bound the *residual gradient norm* to relieve this tension.

Lemma 1. *The gap between the gradient residual norm of the FOCI Unlearning update in Algorithm 1 and a full unlearning update via (4),*

$$||\nabla F(w^-_{Foci}, D')||_2 - ||\nabla F(w^-_{Full}, D')||_2 \quad (16)$$

shrinks as $O(1/n^2)$.

Proof. The full proof is in the appendix. Main idea: Because we only update a subset of parameters, the gradients for the remainder should not change too much. Any change to a selected layer only propagates to other layers by $1/n$, and a Taylor expansion about the new activation for that layer gives the result. □

How L-CODEC achieves acceleration for Unlearning? Sampling with weights proportional to the Lipschitz constant of individual filters/layers is an established approach in optimization, see [17]. We argue that L-CODEC computes an approximation to optimal sampling probabilities. Under a mild assumption that the sampling probabilities have *full* support, it turns out that correctness of our approximate (layer/filter selection) procedure can be guaranteed for unlearning purposes using recently developed optimization tools, see [18]. By adapting results from [17], we can show the following, summarizing the main result of our slice-based unlearning procedure.

Theorem 1. *Assume that layer-wise sampling probabilities are nonzero. Given unlearning parameters* ϵ, δ*, the unlearning procedure in Alg 1 is* (ϵ', δ')*–forgetting where* $\epsilon' > \epsilon, \delta' > \delta$ *represent an arbitrary precision (hyperparameter) required for unlearning. Moreover, iteratively applying our algorithm converges exponentially fast (in expectation) w.r.t. the precision gap, that is, takes (at most)* $O(\log \frac{1}{\mathbf{g}_\epsilon} \log \frac{1}{\mathbf{g}_\delta})$ *iterations to output such a solution where* $\mathbf{g}_\epsilon = \epsilon' - \epsilon > 0, \mathbf{g}_\delta = \delta' - \delta > 0$ *are gap parameters.*

Our result differs from Nesterov's acceleration: we do not use previous iterates in a momentum or ODE-like fashion; rather, here we are closer to primal-dual algorithms where knowing nonzero coordinates at the dual optimal solution can be used to accelerate primal convergence, see [9]. Moreover, since our approach is *randomized*, the dynamics can be better modeled using the SDE framework for unlearning purposes, as in [31]. Here, we do not compute anything extra, although it is feasible for future extensions.

Remark 3. *Our approach to estimate the Lipschitz constant is different from [10] where an SDP must be solved – quite infeasible for unlearning applications. Our approach can be interpreted as solving a simplified form of the SDP proposed there, when appropriate regularity conditions on the feasible set of the SDP are satisfied.*

A note on convexity. Existing methods for guaranteeing removal and performance depend on models being convex. Practical deep learning applications however involve highly nonconvex functions. The intuitions of unlearning for convex problems **directly apply to nonconvex unlearning** with one more technical assumption: minimizers of the learning problem satisfy Second Order Sufficiency (SOS) conditions. SOS guarantees that $\nabla^2 \hat{F}(\hat{w})$, $\hat{H}$ in eq (7) of [28] are PSD, and that the update (8) is an *ascent* direction w.r.t. the loss function on U, making unlearning possible. Guarantees for nonconvex unlearning involve explicitly characterizing a subset of SOS points (so-called "basin of attraction" of population loss), i.e., which points gradient descent can converge to, see §1.3 in [33]. So, will minimizers from first order methods satisfy SOS conditions? Generally, this is not true, e.g., when the Hessian is indefinite, $\hat{H} \not\succeq 0$, the update itself may not be an ascent direction w.r.t. negative of the loss. Here, standard Hessian modification schemes are applicable [35], subsequently using the Newton's step in [30] with a diagonally modified Hessian.

We fix weight decay during training, acting as ℓ_2 regularization and giving us an approximate λ-strong convexity. We also take advantage of this property to smooth our Hessian prior to inversion, intuitively extending the natural linearization about a strongly-convex function. Interestingly, this exactly matches a key conclusion from [2]: weight-decay heavily affects the quality of the measured influence, consistent with our nonconvexity discussion.

Implementation Details. As we only need a subset of the Hessian, we compute the finite difference among the parameters within the blocks selected. For large models, even subsets of model parameters may lead to large Hessian computations, so we move parameters as needed to the CPU for parameter updates. Pairwise distance computations for CI testing via nearest neighbor are carried out on the GPU [37]. Our code although not explicitly optimized achieves reasonable run-time for unlearning for deep models, e.g., one unlearning step for person re-identification task on a ResNet50 model with roughly 24M parameters takes about 3 minutes.

6. L-FOCI in Generic ML Settings

We begin with understanding the value of L-CODEC and L-FOCI for Markov Blanket Identification and progress to applications in typical unlearning tasks involving large neural networks previously infeasible with existing scrubbing tools. See appendix for additional details.

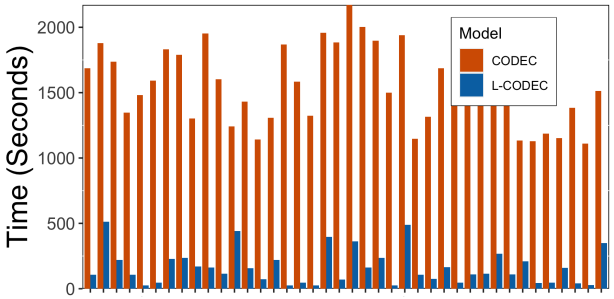


Figure 3. L-CODEC vs CODEC run time comparison for identifying sufficient subsets for each CelebA attribute separately (pairs of columns, details in supplement).

	Raw Data			Feature Maps		
Method	TPR	FPR	Time (s)	TPR	FPR	Time (s)
[36]	0.75	0.50	5124.22	**0.875**	**0.00**	516.19
L-CODEC + CIT	**1.00**	**0.50**	**402.10**	0.75	**0.00**	117.29
L-CODEC + L-FOCI	N/A			0.833	0.50	**0.464**

Table 1. 3D-Bullseye Markov Blanket identification. CIT represents the model in [36]. Both L-CODEC and L-FOCI run much faster than recent Markov Blanket identification schemes. L-FOCI is not applicable to the multi-dimensional raw data setting.

L-CODEC Evaluation. To assess speedup gained in the discrete setting when running L-CODEC, we construct the Markov Blanket for specific attributes provided as side information with the CelebA dataset. Fig. 3 shows the wall-clock times for Markov Blanket Selection via FOCI and L-FOCI for each attribute.

Markov Blanket Identification. We replicate the experimental setup in Sec 5.3 of [36], where a high dimensional distribution over a ground truth graph is generated, and feature mappings are used to reduce the dimension and map to a latent space. Table 1 summarizes subset identification efficacy and runtime. Replacing conditional mutual information (CMI) with L-CODEC, we see a clear improvement in both runtime and Markov Blanket identification over the raw data, and comparable results in the latent feature space. Using L-FOCI directly in the feature space, we identify an additional spurious feature not part of the Markov Blanket, but runtime is significantly faster.

Spurious Feature Regularization. This Markov Blanket (MB) identification scheme can be used to address spurious feature effects on traditional NN models. A straightforward approach would be to directly add a loss term for each potentially important feature over which we would like to regularize, $\mathcal{L}(\theta) + \sum_{S \in \mathcal{S}} R_S(\theta)$. However, with a large number of outside factors S, this can adversely effect training. We instead use L-FOCI to identify the set of minimal factors that, when conditioned, make the rest conditionally independent. Then it is only necessary to include regularizers over $S \in MB(Y)$.

We evaluate a simple attribute image classification set-

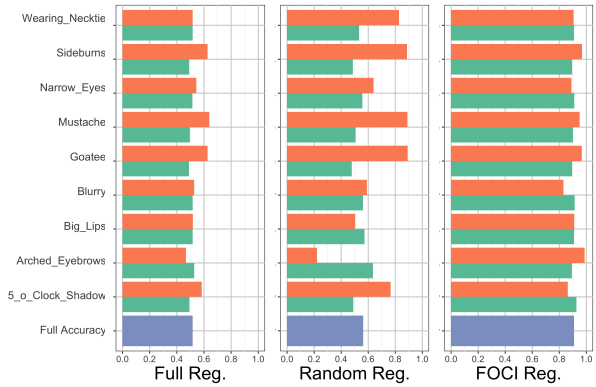


Figure 4. Validation accuracies after training to predict "No Beard" in the CelebA dataset. (L to R) regularization for all features, for a random subset, and via FOCI. Green indicates accuracy on the data with that feature, red, without.

ting using the CelebA dataset. We run L-FOCI over the attributes as in our L-CODEC evaluation, and regularize using a Gradient Reversal Layer for a simple accuracy term over those attributes. Results in Fig. 4 clearly show that selection with FOCI provides the best result, maintaining high overall accuracy but also preserving high accuracy on sets of samples with/without correlated attributes.

7. L-FOCI for Machine Unlearning

7.1. Compare to Full Hessian Computation

For simple regressors, we can compute the full Hessian and compare results generated by a traditional unlearning update, our L-FOCI update, and a random selection update. To reduce variance and show the best possible random selection, we run our L-FOCI and randomly choose a set of the same size for each random selection. Fig. 5 (left) shows validation and residual accuracies for 1000 random removals from MNIST (average over 10 runs).

Are we selecting reasonable subsets? A natural question is whether the subset selection via L-FOCI is any better than random, given that we are effectively taking a smaller global step. We answer this in the affirmative with a simple comparison with a random selection of size equal to the set selected by L-FOCI. Fig. 5 (left) shows that the sample gradient norm for selections made by L-FOCI are larger than those of a random selection: the subset of the model scrubbed of this specific sample has a larger impact on its final loss, and thus the gradient norm post-removal is large.

Does the formulation scale? We scrub random samples from various CIFAR-10 models, and evaluate performance for the same set of hyperparameters. When the models are larger than logistic regression, it is infeasible to estimate the full Hessians, so we *must* use our L-FOCI selection update. Fig. 5 (right) shows removal performance over many typical models with varying sizes. Models that have higher base accuracies tend to support more removals before performance drops. This matches results for differentially private models: models that generalize well may not have overfit and thus may already be private, allowing "fast" forgetting.

Tradeoff vs Retraining. While our focus is the setting in which retraining is not feasible, where we can retrain we compare validation accuracies as a function of number of removals. Using a subset of MNIST, we train to convergence and iteratively remove samples using our construction, retraining fully at each step for comparison. With 1000 training samples from each class and reasonable settings of privacy parameters ($\epsilon = 0.1, \delta = 0.01$), we support a large percentage of removals until validation accuracy drops more than a few percent, see Fig. 6.

7.2. Removal in NLP models

We now scrub samples from transformer based models using LEDGAR [34], a multilabel corpus of legal provisions in contracts. We use the prototypical subset which contains 110156 provisions pertaining to 13 most commonly used labels based on frequency. Our model is a fine-tuned DistilBERT [29] and uses the $[CLS]$ token as an input to the classification head. Table. 7b shows results of scrubbing the provisions from two different classes; *Governing Laws* and *Terminations* which have the highest/lowest support in the test set. As expected with increasing ϵ, i.e., lower privacy guarantees, we can support more number of removals based on the Micro F1 score of the overall model. The Micro F1 scores, for the removed class fall off rapidly, while the change in overall scores is more gradual.

7.3. Removal from Pretrained Models

The above settings show settings where a sample from one specific source may be removed. A more direct application of unlearning is completely removing samples from a specific class; a compelling use case is face recognition.

We utilize the VGGFace dataset and model, pretrained from the original work in [21, 27]. The model uses a total of approximately 1 million images to predict the identity of 2622 celebrities in the dataset. Using a reconstructed subset of 100 images from each person, we first fine-tune the model on this subset for 5 epochs, and use the resultant models as estimates of the Hessian. In this setting, the VGGFace model is very large, including a linear layer of size 25088×4096. Selecting even a few slices from this layer results in a Hessian matrix unable to fit in typical memory. For this reason, we run a "cheap" version of L-FOCI: we select only one slice that results in the largest conditional dependence on the output loss.

Fig. 7a show results for scrubbing consecutive images from one individual in the dataset for a strong privacy guarantee of $\epsilon = 10^{-5}$. As the number of samples scrubbed increases, the performance on that class drops faster than on the residual set, exactly as desired.

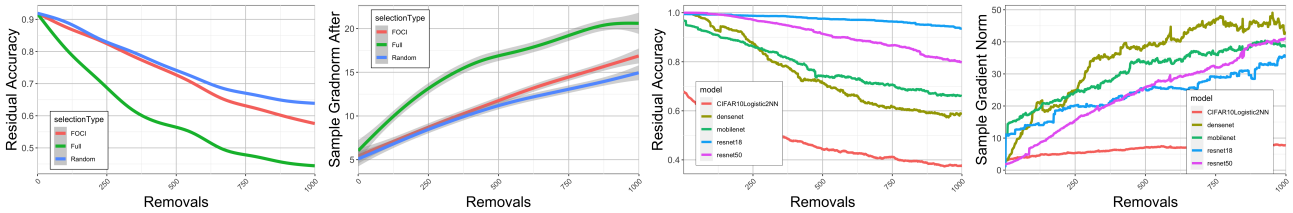


Figure 5. (Left) Residual Accuracies & Sample Gradient Norm of removal for an MNIST Logistic Regressor. Averaged over 10 runs. (Right) Residual accuracies and sample gradient norms for various CIFAR-10 models.

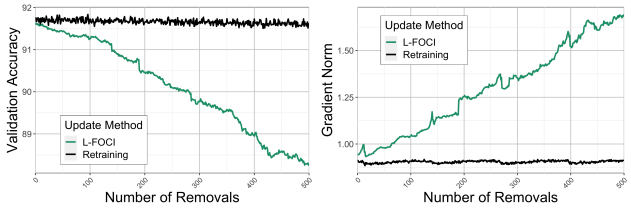
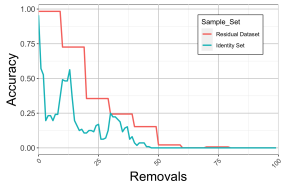


Figure 6. MNIST Retraining comparison averaged over 8 runs. Validation accuracies and residual gradient norms.



ϵ	# Supported Removals	
	Governing Laws	Terminations
0.1	> 100	> 100
0.01	> 100	> 100
0.001	18	21
0.0005	6	7

Figure 7. (Left) Scrubbed and Residual Accuracies (every 10 removals) for $\epsilon = 1e^{-5}$. The accuracy drop for the residual set is gradual up to a certain number of removals. (Right) Scrubbing transformer model for provision classification.

7.4. Removal from Person re-identification model

As a natural extension to our experiments on face recognition, we evaluate unlearning of deep neural networks trained for person re-identification. Here, the task is to associate the images pertaining to a particular individual but collected in diverse camera settings, both belonging to the same camera or from multiple cameras. In our experiments, we use the Market-1501 dataset [38] and a Resnet50 architecture which was trained for the task. We unlearn samples belonging to a particular person, one at a time, and check the performance of the model. Experimental results are in agreement with results reported for the transformer model as well as the VGGFace model. With very small values of ϵ i.e. 0.0005 the number of supported removals is limited to less than 10 depending on the person id being removed. However, with a larger value of ϵ, e.g., 0.1, all potential samples can be removed without a noticeable degradation in model performance in terms of mAP scores. In Fig. 8, we clearly see that after scrubbing a model for a particular person, its predictions for that particular individual become meaningless whereas the predictions on other

Figure 8. Activation maps from a model scrubbed for the person on the left (right set is not scrubbed). For each triplet, from (L to R) are the original image, the activation map and its image overlay. Note the effect of scrubbing: activations change significantly for the scrubbed sample (compare column 2 to 3) whereas remain stable for the non-scrubbed sample (compare column 5 to 6).

classes are still possible with confidence, as desired. Additional experiments with different datasets, model architectures and other ablations for deep unlearning for person re-identification models are presented in the appendix.

8. Conclusion

Our selection scheme identifies a subset of parameters to update and significantly reduces compute requirements for standard Hessian unlearning. For smaller networks with a large number of removals, retraining may be effective, but when full training sets are not available or retraining is costly, unlearning in some form is needed. We show the ability to approximately unlearn for large models prevalent in vision, a capability that has not so far been demonstrated.

Social Impact. Indiscriminate use of personal data in training large AI models is ethically questionable and sometimes illegal. We need mechanisms to ensure that AI models operate within boundaries specified by society and legal guardrails. As opt-out laws get implemented, compliance on the service-provider end will entail costs. While our contributions cannot guarantee perfect forgetting, with additional validation they can become a part of a suite of methods for unlearning.

Acknowledgments. This work was supported by NIH grants RF1AG059312, RF1AG062336 and RF1AG059869, NSF award CCF 1918211 and funds from the American Family Insurance Data Science Institute at UW-Madison. Sathya Ravi was supported by UIC-ICR start-up funds.

References

[1] Mona Azadkia and Sourav Chatterjee. A simple measure of conditional dependence. *arXiv preprint arXiv:1910.12327*, 2019. 3, 4

[2] Samyadeep Basu, Phil Pope, and Soheil Feizi. Influence functions in deep learning are fragile. In *International Conference on Learning Representations*, 2021. 6

[3] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6541–6549, 2017. 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. IEEE, 2021. 3

[5] 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. 3

[6] Nicholas Carlini, Samuel Deng, Sanjam Garg, Somesh Jha, Saeed Mahloujifar, Mohammad Mahmoody, Shuang Song, Abhradeep Thakurta, and Florian Tramer. An attack on instahide: Is private learning possible with instance encoding? *arXiv preprint arXiv:2011.05315*, 2020. 1

[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, 2019. 1

[8] Sourav Chatterjee. A new coefficient of correlation. *Journal of the American Statistical Association*, 0(0):1–21, 2020. 4

[9] Jelena Diakonikolas and Lorenzo Orecchia. The approximate duality gap technique: A unified theory of first-order methods. *SIAM Journal on Optimization*, 29(1):660–689, 2019. 5

[10] Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George J Pappas. Efficient and accurate estimation of lipschitz constants for deep neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 6

[11] Ruth Fong and Andrea Vedaldi. Net2vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8730–8738, 2018. 2

[12] FTC. California company settles ftc allegations it deceived consumers about use of facial recognition in photo storage app, Jan 2021. 1

[13] A Ginart, M Guan, G Valiant, and J Zou. Making ai forget you: Data deletion in machine learning. *Advances in neural information processing systems*, 2019. 2, 3

[14] Aditya Golatkar, Alessandro Achille, Avinash Ravichandran, Marzia Polito, and Stefano Soatto. Mixed-privacy forgetting in deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 792–801, June 2021. 2, 3

[15] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9304–9312, 2020. 3

[16] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Forgetting outside the box: Scrubbing deep networks of information accessible from input-output observations. In *European Conference on Computer Vision*, pages 383–398. Springer, 2020. 2, 3

[17] Eduard Gorbunov, Filip Hanzely, and Peter Richtárik. A unified theory of sgd: Variance reduction, sampling, quantization and coordinate descent. In *International Conference on Artificial Intelligence and Statistics*, pages 680–690. PMLR, 2020. 5

[18] Robert Mansel Gower, Nicolas Loizou, Xun Qian, Alibek Sailanbayev, Egor Shulgin, and Peter Richtárik. Sgd: General analysis and improved rates. In *International Conference on Machine Learning*, pages 5200–5209. PMLR, 2019. 5

[19] Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. Certified data removal from machine learning models. In *International Conference on Machine Learning*, pages 3832–3842. PMLR, 2020. 3

[20] Jules. Harvey, Adam. LaPlace. Exposing.ai, 2021. 1

[21] Gary B Huang, Marwan Mattar, Tamara Berg, and Eric Learned-Miller. Labeled faces in the wild: A database forstudying face recognition in unconstrained environments. In *Workshop on faces in'Real-Life'Images: detection, alignment, and recognition*, 2008. 7

[22] Zachary Izzo, Mary Anne Smart, Kamalika Chaudhuri, and James Zou. Approximate data deletion from machine learning models. In *International Conference on Artificial Intelligence and Statistics*, pages 2008–2016. PMLR, 2021. 3

[23] Masayuki Karasuyama and Ichiro Takeuchi. Multiple incremental decremental learning of support vector machines. *Advances in neural information processing systems*, 22:907–915, 2009. 3

[24] Kate Kaye. The ftc's new enforcement weapon spells death for algorithms, Mar 2022. 1

[25] Seth Neel, Aaron Roth, and Saeed Sharifi-Malvajerdi. Descent-to-delete: Gradient-based methods for machine unlearning. In *Algorithmic Learning Theory*, pages 931–962. PMLR, 2021. 3

[26] Peter Orbanz. Probability theory, Spring 2016. 4

[27] Omkar M. Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. In *British Machine Vision Conference*, 2015. 7

[28] Enrique Romero, Ignacio Barrio, and Lluís Belanche. Incremental and decremental learning for linear support vector machines. In *International Conference on Artificial Neural Networks*, pages 209–218. Springer, 2007. 3

[29] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019. 7

[30] Ayush Sekhari, Jayadev Acharya, Gautam Kamath, and Ananda Theertha Suresh. Remember what you want to forget: Algorithms for machine unlearning, 2021. 2, 5, 6

[31] Umut Simsekli, Lingjiong Zhu, Yee Whye Teh, and Mert Gurbuzbalaban. Fractional underdamped langevin dynamics: Retargeting sgd with momentum under heavy-tailed gradient noise. In *International Conference on Machine Learning*, pages 8970–8980. PMLR, 2020. 5

[32] Yiyou Sun, Sathya N. Ravi, and Vikas Singh. Adaptive activation thresholding: Dynamic routing type behavior for interpretability in convolutional neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 2

[33] Yann Traonmilin and Jean-François Aujol. The basins of attraction of the global minimizers of the non-convex sparse spike estimation problem. *Inverse Problems*, 36(4):045003, feb 2020. 6

[34] Don Tuggener, Pius von Däniken, Thomas Peetz, and Mark Cieliebak. Ledgar: a large-scale multi-label corpus for text classification of legal provisions in contracts. In *12th Language Resources and Evaluation Conference (LREC) 2020*, pages 1228–1234. European Language Resources Association, 2020. 7

[35] Stephen Wright, Jorge Nocedal, et al. Numerical optimization. *Springer Science*, 35(67-68):7, 1999. 6

[36] Alan Yang, AmirEmad Ghassami, Maxim Raginsky, Negar Kiyavash, and Elyse Rosenbaum. Model-augmented conditional mutual information estimation for feature selection. In *Conference on Uncertainty in Artificial Intelligence*, pages 1139–1148. PMLR, 2020. 4, 6

[37] Zhanpeng Zeng, Yunyang Xiong, Sathya Ravi, Shailesh Acharya, Glenn M Fung, and Vikas Singh. You only sample (almost) once: Linear cost self-attention via bernoulli sampling. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12321–12332. PMLR, 18–24 Jul 2021. 6

[38] Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. Scalable person re-identification: A benchmark. In *Proceedings of the IEEE international conference on computer vision*, pages 1116–1124, 2015. 8