Deep Unlearning via Randomized Conditionally Independent Hessians
Supplementary Material

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
https://github.com/vsingh-group/LCODEC-deep-unlearning/

Contents

1 Theoretical Results **2**
1.1 Proof of Lemma 1 . . . 2
1.2 Proof of Theorem 1 . . . 3

2 Experimental Details **4**
2.1 Markov Blanket Selection . . . 4
2.2 L-CODEC vs CODEC Speed Results . . . 4
2.3 MNIST Toy Results . . . 4
2.4 Retraining Comparisons . . . 4
MNIST: Affects of l_2 Regularization and Weight Decay . . . 4
CIFAR Retraining Comparisons: A Note on Batch Normalization . . . 5
2.5 CIFAR-10 Model Comparisons . . . 6
2.6 LEDGAR DistillBERT Details . . . 6
2.7 VGG-Face Identification Scrubbing . . . 6
2.8 Person Re-identification . . . 7

3 Conditional Independence and Parameter Selection via L-CODEC **7**

4 Alternate Hessian Approximations **7**

[*]Joint First Authors.

1 Theoretical Results

1.1 Proof of Lemma 1

Let us take D to be the training set; w.l.o.g. z is the point being removed. Let the residual dataset be $D' = D \setminus z$. Denote w^-_{Full} as the weight parameters after doing a full Hessian update and w^-_{Foci} as the weight parameters after doing a FOCI selected Hessian update. In an ideal case, we want (w^-_{Foci}, D')/(w^-_{Full}, D') to be as close as possible to (w^*, D'). Note that we consider both (w^*, D) and (w^*, D') to be 0 as we don't expect model parameters to change drastically for one sample once trained to convergence.

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

$$||\nabla\mathcal{L}(w^-_{Foci}, D')||_2 - ||\nabla\mathcal{L}(w^-_{Full}, D')||_2 \tag{1}$$

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

Proof. Let w to be a network of many linear layers with possible activation functions; we can think of the norm as the sum of norm of gradients for each layer. Hence, for any model parameters w and dataset D, we have:

$$||\nabla\mathcal{L}(w, D)||_2 \coloneqq \sum_{l\in L} ||\nabla\mathcal{L}(w_l, D)||_2 \tag{2}$$

FOCI identifies a subset $T \subset L$ slices or layers that are to be updated. Let $R = L \setminus T$ be the remainder of the network which is not updated. Hence, 2 for (w^-_{Foci}, D') can be written as:

$$\begin{aligned}
||\nabla\mathcal{L}(w^-_{Foci}, D')||_2 &\coloneqq \sum_{l\in L} ||\nabla\mathcal{L}(w^-_{Foci_l}, D')||_2 && (3)\\
&= \sum_{l\in T} ||\nabla\mathcal{L}(w^-_{Foci_l}, D')||_2 + \sum_{l\in R} ||\nabla\mathcal{L}(w^-_{Foci_l}, D')||_2 && (4)\\
&= \sum_{l\in T} ||\nabla\mathcal{L}(w^-_{Foci_l}, D')||_2 + \sum_{l\in R} ||\nabla\mathcal{L}(w^*_l, D')||_2 && (5)
\end{aligned}$$

The last line follows from the fact that layers in R are not updated.

We will next show how for the remainder of the dataset D', the changes in T propagate minimally when there are a large number of data points, n in the training set.

W.L.O.G. assume that we have a 3 layer network with the form:

$$(L_3(L_2(L_1(x))) \tag{6}$$

For the point being removed $z := (x, y)$; let L_2 be the intermediate layer which is selected for update by FOCI. Before the update, activations out of L_2 are of the form $a_2 = L_2(L_1(x)) = L_2(a_1)$. After the update, activations out of L_2 can be written as:

$$\begin{aligned}
a'_2 = L'_2(L_1(x)) &= L'_2(a_1) && (7)\\
&= w'_2 a_1 && (8)\\
&= (w_2 + \delta_{w_2})a_1 && (9)\\
&= w_2 a_1 + \delta_{w_2} a_1 && (10)\\
&= a_2 + \delta_{w_2} a_1 && (11)
\end{aligned}$$

The Second line follows because L_1 isn't updated. For the following layer L_3, we have $a_3 = L_3(a_2)$ before the update. After,

$$
\begin{aligned}
a_3' &= L_3(a_2') && (12)\\
&= L_3(a_2 + \delta_{w_2} a_1) && (13)\\
&= L_3(a_2) + \nabla L_3(a_2)\delta_{w_2} a_1 + \mathcal{O}((\delta_{w_2} a_1)^2) && (14)\\
&= L_3(a_2) + 0 + \mathcal{O}((\delta_{w_2} a_1)^2) && (15)
\end{aligned}
$$

The first-order term goes to zero, as L_3 has not been updated and we assume full model convergence.

For the [12] update.

$$\delta_{w_2} = \frac{1}{(n-1)}(\hat{H}^{-1}) \sum_{z \in \{(x_k, y_k)\}} \nabla f(\hat{w}, z) \tag{16}$$

Hence, $\delta_{w_2}^2 \propto \frac{1}{n^2}$. Therefore, for large values of n, the third term in the equation above approaches 0. So, $a_3' = L_3(a_2)$. This shows that propagation is minimal. Similar arguments regarding null space for over-parameterized deep networks have been mentioned in [3].

Now, looking back at the residual gradient norm, we have:

$$||\nabla \mathcal{L}(w^-_{Foci}, D')||_2 = \sum_{l \in T} ||\nabla \mathcal{L}(w^-_{Foci_l}, D')||_2 + \sum_{l \in R} ||\nabla \mathcal{L}(w_l^*, D')||_2 \tag{17}$$

Based on the above argument of minimal propagation, the second term above goes to 0 for layers/slices in R. Therefore,

$$||\nabla \mathcal{L}(w^-_{Foci}, D')||_2 = \sum_{l \in T} ||\nabla \mathcal{L}(w_l^-, D')||_2 \tag{18}$$

and as such the gap between this and the full update is only the difference on the set R, shrinking as $O(1/n^2)$. □

1.2 Proof of Theorem 1

Theorem 1. *Assume that layer-wise sampling probabilities are nonzero. Given (user specified) unlearning parameters ϵ, δ, the unlearning procedure in Algorithm 1 in the main paper is $(\epsilon', \delta')-$forgetting where $\epsilon' > \epsilon, \delta' > \delta$ represent an arbitrary precision (hyperparameter) required for unlearning. Moreover, iteratively applying our algorithm converges exponentially fast (in expectation) with respect to 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.*

Proof. Our proof strategy is to show that our update step in Algorithm 1 is a specific form of Randomized Block Coordinate Descent (R-BCD) method. Then, we simply apply existing convergence rates of RBCD for general smooth minimization problems. In particular, our method can be seen as an extension of SEGA method in Corallary A.7. [4] where the descent direction is provided by using inexact inverse hessian metric [6]. The key difference in our setup is that the sampling probabilities are computed using the CODEC procedure instead of the random sampling at each step. We make the following three observation in our setup that immediately asserts correctness of the procedure.

First, by our construction in equation (11) in the main paper, the sampling probabilities have full support. That is, the probability of selecting a particular weight in the neural network is strictly positive since $\xi \sim \mathcal{N}(0, \sigma^2), \sigma > 0$ is a continuous distribution which has unbounded support. Second, the overall rate of speed of convergence depends on the condition number of the (fixed) Hessian at the optimal solution since exact (ϵ, δ) unlearning is equivalent to linear least squares problem. Third, our update step is equivalent to a projected (or sketched) primal step, see equation 13 in (ArXiv Version [7]). From these observations, we can see that our overall method is equivalent to SEGA in [4] or its noisy extension since we use only a small set of samples (to be unlearned) at each iteration. Consequently, we obtain the deterministic geometric rate of convergence (in expectation) by applying Corollary A.8. where σ in their paper corresponds to the $\epsilon' - \epsilon > 0$ gap in our setup. Now, to get the probabilistic ϵ', δ' unlearning guarantee for the solution presented by our algorithm, we use Lemma 10 in [12] on the solution returned, completing our proof. □

2 Experimental Details

Experiments were conducted using PyTorch 1.8 and CUDA Toolkit 10.2, run on Nvidia 2080 TIs and individual Nvidia A100s. Parallelization only occurred across runs; the attached code can be run with any CUDA/PyTorch setup with the appropriate dependencies.

2.1 Markov Blanket Selection

Experimental settings were taken from [16], with code adapted from https://github.com/syanga/model-augmented-mutual-information. 5000 samples were used for generating the data, and 100 trials/perumutations were conducted for the CIT testing framework.

2.2 L-CODEC vs CODEC Speed Results

The full figure can be seen below, at Figure 1. Each pair of columns corresponds to a different attribute in the CelebA dataset as the focal variable of interest, with the rest as potential conditioning variables.

2.3 MNIST Toy Results

Training for MNIST Logistic Regressor models was run using SGD with a learning rate of 0.1, batch size of 256, and weight decay of 0.01 for 50 epochs. 1000 perturbations were used for distribution approximation. Privacy parameters were set to $\epsilon = 0.1, \delta = 0.01$. Figures and numbers in the main paper were averaged over 10 replications, for a random choice of 1000 samples to scrub.

2.4 Retraining Comparisons

MNIST: Affects of l_2 Regularization and Weight Decay

Repeating the retraining comparison in the main paper with a larger regularization, we see that the effects of removal are significantly diminished and the model can support a larger number of removals before large performance drops.

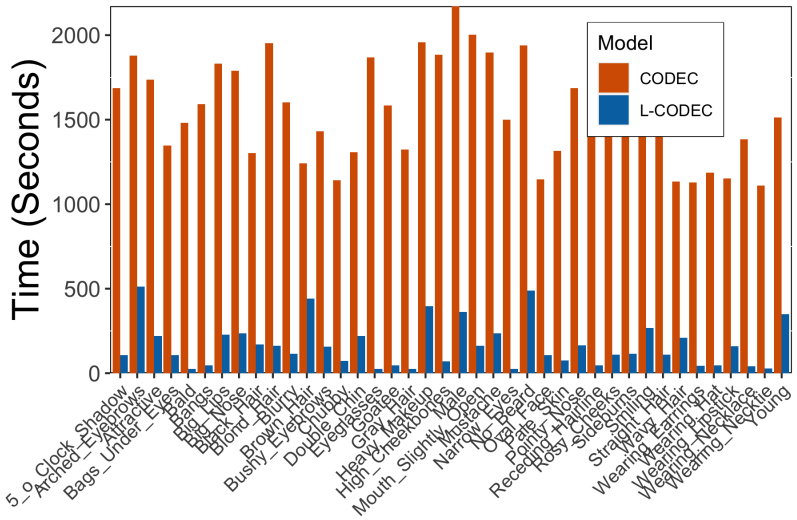


Figure 1: L-CODEC vs CODEC run time comparison for identifying sufficient subsets for each CelebA attribute separately (pairs of columns, details in supplement).

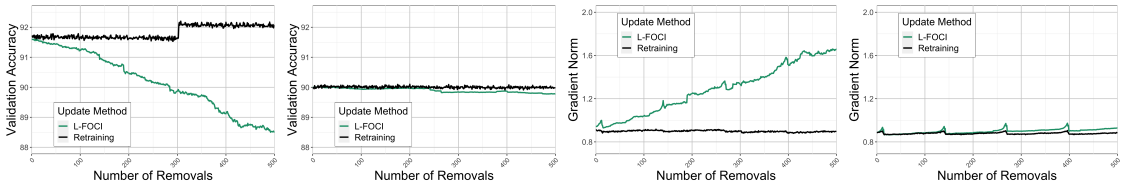


Figure 2: MNIST Retraining results, comparing the effect of weight decay on unlearning via our LFOCI unlearning scheme and retraining.

CIFAR Retraining Comparisons: A Note on Batch Normalization

An important requirement for our retraining experiment is that our residual training set used for both scrubbing validation and retraining is able to take on any size, including 1 and any size for which the modulus over the batch size equals 1. This causes particular problems when models include batch normalization layers: general practice in training deep neural networks includes the choice of dropping the last batch, so as to avoid issues of unbalanced batch sizes. For our setting we *cannot* drop these batches, because we explicitly want to measure and compute on networks trained with and without specific samples. While we can "skip" removals during our experimentation, this can still lead to odd behavior, see Figure 3. The spikes are exactly congruent with points in the removal process corresponding to a final batch size of 1 for retraining. In general, care must be taken when attempting to unlearn from batchnorm models, and further work may be necessary to adequately address it, both in theory and practice.

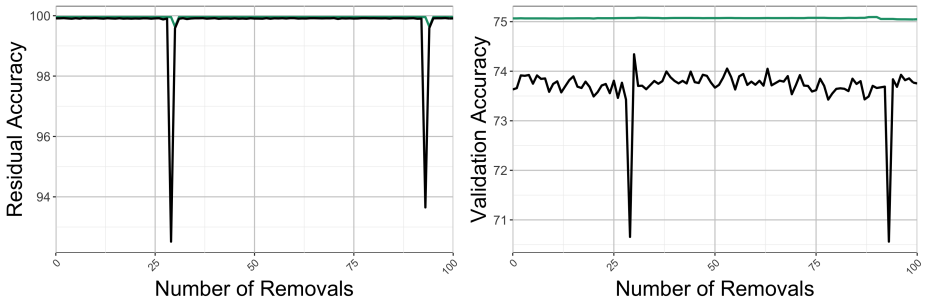


Figure 3: Retraining Results on CIFAR. Dips occur at removal counts where the modulus equals 1.

2.5 CIFAR-10 Model Comparisons

Models were trained using Torch Hub, with a batch size of 64, learning rate of 0.1 for all models except VGG-11/bn, for which 0.01 was used. Data augmentation was NOT used, and weight decay was set to 0.01. 1000 perturbations were used for distribution approximation. Privacy parameters were set to $\epsilon = 0.1, \delta = 0.01$. Figures and numbers in the main paper were averaged over 2 replications, for a random choice of 1000 samples to scrub.

2.6 LEDGAR DistillBERT Details

For the NLP experiments, we used a pretrained model from HuggingFace as a starting point. Specifically, we used the transformer model "distilbert-base-uncased", https://huggingface.co/distilbert-base-uncased which is a distilled version of the BERT base model, smaller and faster than BERT. It was pretrained on the same corpus in a self-supervised fashion, using the BERT base model as a teacher. DistilBERT [11] was pre-trained on the raw texts only, without any human labels. The three losses used for the pre-training are that of distillation loss, masked language modelling and cosine embedding loss. This pre-trained model was then fine-tuned for the downstream task of provision classification using the LEDGAR dataset introduced in [14]. We used the prototypical dataset which had 13 most common labels based on frequency. The model was fine-tuned for 4 epochs, updating all of it's parameters without any freezing based on binary cross entropy loss with class weighting. The labels were converted to one-hot vectors and hence binary cross entropy loss was used. Learning rate used was $5e^{-5}$ and weight decay of 0.01. No weight decay was applied for bias and normalization layer parameters. We used batch size of 256 and restricted the maximum length of tokens to 128 per data point. Further gradients were clipped based on the infinity norm to a value of 1.0. We used AdamW optimizer with an epsilon value of $1e^{-8}$; and the learning rate scheduler used was WarmupLinearSchedule both from PyTorch_Transformers.

For unlearning experiments on this model, we remove provisions pertaining to a specific class. We removed samples from two classes namely "Governing Laws" and "Terminations" which had the highest and lowest support respectively. We were able to removed a varying number of samples from these classes based on the selection of the privacy parameter of ϵ for scrubbing. The results are tabulated in the main paper.

2.7 VGG-Face Identification Scrubbing

For this setting, the trained model was downloaded from https://www.robots.ox.ac.uk/~vgg/data/vgg_face/ and converted to PyTorch via https://github.com/prlz77/vgg-face.pytorch. A partial version of the dataset

was constructed using the list of image URLs, consisting of 100 images for each identity within the set. The images were processed as described in the original paper [10].

Fine tuning was done for 4 epochs to estimate the Hessian for the sample downloaded using SGD with a learning rate of 0.0001 and a weight decay/l_2 regularization of 0.01, with a batch size of 16.

For unlearning, 100 images for a specific identity were randomly ordered and removed with $\epsilon = 0.0001, \delta = 0.01$. 100 perturbations were used to estimate the activation and loss distributions for L-FOCI.

2.8 Person Re-identification

We discuss in more detail the experimental details of unlearning deep neural networks for the person re-identification task in this section. We consider four different datasets namely, Market1501 [17], MSMT17 [15], PRID [5] and QMUL GRID [8]. We unlearn from different deep neural networks including ResNet50, a variant of ResNet50 with a fully connected layer (called Resnet50_fc512), Multi-Level Factorisation Net (MLFN) and MobilNet_V2. In all cases the models were first trained to reasonable accuracy as per benchmarks before proceeding with unlearning a randomly selected individual's identity from the corresponding dataset. To perform experiments pertaining to person re-identification we make use of the popular framework torchreid [18]. We had to make changes to the original code in order to make our procedure function correctly in this framework. We include this modified source within the code presented in the supplement. We use Adam as the optimizer, a step scheduler and learning rate of 0.0003 across all person re-identification datasets and models used. We use softmax loss and weights were initialized using a model pre-trained on ImageNet in all cases. Images were resized to 256×128 before being used as input to any of the models. The number of training epochs was chosen accordingly to allow the training to have converged. Results from multiple runs involving different models, datasets and the privacy parameter (ϵ) are conclusive. With lower value of ϵ, e.g. 0.0005, the number of samples that could be unlearned for a particular class while maintaining model performance was lower than what could be unlearned for a higher value of the privacy parameter ϵ, e.g. 0.1. For the smaller datasets, i.e. PRID and QMUL GRID, which have approximately 2 samples per class, the unlearning procedure lead to more drastic changes as expected and it could be observed that our selection procedure selected many more layers to update than what it did for the larger datasets. Activation maps from some experiments are presented in Fig 4.

3 Conditional Independence and Parameter Selection via L-CODEC

Our algorithm is directly adapted from [1] to our parameter selection setting. Algorithm 1 shows the procedure, described for arbitrary random variables in Section 5 of [1].

While tests for independence exist, CODEC directly estimates explanation of variance with and without the conditional variable(s) of interest. For readers interested in conditional independence more generally, and statistical and theoretical foundations, please see [13, 2]. More recent information-based formulations can be found in [16] and references therein.

4 Alternate Hessian Approximations

Typical approximations are non often non-sparse; a key focus of our proposal is a reasonably informed sparse estimation in deep unlearning: we cannot allocate both full networks and the space for an inverse for 50K$+$ parameters (needs $10+$GB alone). For Deep unlearning specifically, our sub selection makes this possible. Diagonal modification still needs full parameter updates. However, we explored the utilization of

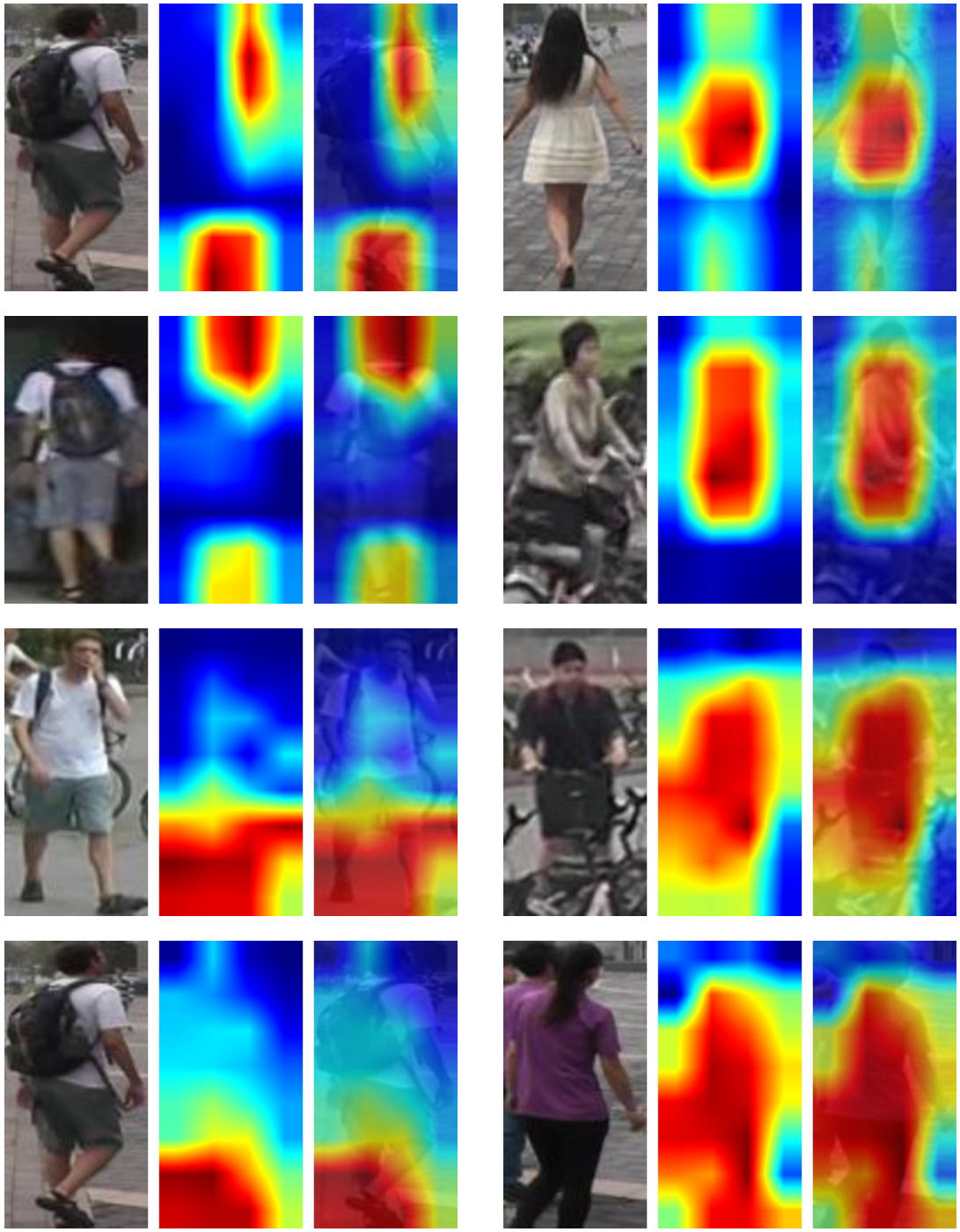
Algorithm 1: Parameter MB Identification via L-CODEC (L-FOCI)

Data: z', the full parameter set w indexed by $\Theta := \{1, \ldots, d\}$
Result: Sufficient set $P \subseteq \Theta$
Identify $p \in \Theta$ that maximizes $T(z', w_p)$
Set $P = \{p\}$
while $T(z', w_{\Theta \setminus P} | w_P) > 0$ **do**
  Identify $p \in \Theta \setminus P$ that maximizes $T(z', w_{\Theta \setminus P} | w_{\Theta \cup s})$
  if $T(z', w_{\setminus P}, w_{P \cup p}) < 0$ **then**
    break
  else
    Append $P = P \cup p$
  end
end

other Hessian inverse approximation schemes. More specifically, we implemented an unlearning scheme based on Kronecker-Factored Approximate Curvature (K-FAC) [9] which exploits an efficient invertible approximation of a deep learning model's Fisher information matrix which can be non-sparse and neither low rank nor diagonal. In an experimental setup, we perform unlearning based on K-FAC from an multi layer perceptron model trained on MNIST dataset. We don't see any observable updates happening to the model based on validation metrics. Whereas, the exact same model with the exact same set of parameters can unlearn the same set of data-points using our proposed deep unlearning method based on LCODEC. We would like to point out that in order to unlearn from deep models using existing approximations schemes like K-FAC, we might have to re-imagine the update step. This demands further investigation. In other words our procedure may not be more broadly applicable to non-sparse general Hessian inverse approximations without an obvious CI structure.

We have included the code to compare K-FAC based unlearning with LCODEC based unlearning in the file https://github.com/vsingh-group/LCODEC-deep-unlearning/blob/main/scrub/kfac_scrub.py

In our implementation we heavily rely on the KFAC approximations of the Hessian as provided in https://github.com/cybertronai/autograd-lib . More instructions can be found in the README.



(a) Scrubbed class (b) Unscrubbed class

Figure 4: Activation maps from a different models (top two rows correspond to MLFN and bottom two correspond to MobileNet_V2; both trained on Market_1501) 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).

References

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

[2] A Philip Dawid. Conditional independence in statistical theory. *Journal of the Royal Statistical Society: Series B (Methodological)*, 41(1):1–15, 1979.

[3] 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.

[4] 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] Martin Hirzer, Csaba Beleznai, Peter M. Roth, and Horst Bischof. Person Re-Identification by Descriptive and Discriminative Classification. In *Proc. Scandinavian Conference on Image Analysis (SCIA)*, 2011.

[6] Nicolas Loizou and Peter Richtárik. Convergence analysis of inexact randomized iterative methods. *SIAM Journal on Scientific Computing*, 42(6):A3979–A4016, 2020.

[7] Nicolas Loizou and Peter Richtárik. Convergence analysis of inexact randomized iterative methods, 2019.

[8] Chen Change Loy, Tao Xiang, and Shaogang Gong. Multi-camera activity correlation analysis. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1988–1995. IEEE, 2009.

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

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

[11] 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.

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

[13] Peter Spirtes, Clark N Glymour, Richard Scheines, and David Heckerman. *Causation, prediction, and search*. MIT press, 2000.

[14] 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.

[15] Longhui Wei, Shiliang Zhang, Wen Gao, and Qi Tian. Person transfer gan to bridge domain gap for person re-identification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 79–88, 2018.

[16] 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.

[17] 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.

[18] Kaiyang Zhou, Yongxin Yang, Andrea Cavallaro, and Tao Xiang. Omni-scale feature learning for person re-identification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3702–3712, 2019.