ECE-GY 7123
Deep Learning Major Project
Data Retraction and Ensuring Model Integrity

Utkarsh Prakash Srivastava, Srushti Jagtap, Raghav Rawat
ups2006, sj4182, rr3418
Department of Electrical and Computer Engineering
Tandon School of Engineering,
New York University, NY, USA
Github Link : https://github.com/utkarsh231/DL-Major-Project

Abstract

Removing sensitive information from machine learning models is crucial in scenarios where data leaks or privacy issues arise, such as when sensitive data like credit card numbers inadvertently enter the model. In the domain of computer vision, poisoning attacks pose a significant threat, where adversaries manipulate training data labels to deceive object recognition tasks. Existing methods for unlearning in machine learning focus on removing individual data points but struggle with scalability when larger groups of features and labels need to be reverted.

In this project, we used a method for unlearning features and labels simultaneously. Leveraging influence functions, this approach enables closed-form updates of model parameters, allowing retrospective adaptation of training data influence on the model. Our method addresses data leaks and privacy issues by correcting defects introduced by poisoning attacks, significantly improving the robustness and privacy of machine learning models.

Introduction

Machine learning has revolutionized the analysis of personal data and the development of data-driven services. However, the widespread use of learning models has raised concerns about privacy, as these models can inadvertently capture sensitive information from the training data and expose it to users. Privacy regulations such as the European GDPR [Parliament and Council 2016] provide users with the right to request the removal of their personal data from these models, known as the "right to be forgotten." However, deleting data from a learning model is a challenging task, often requiring selective reverting of the learning process. Without specific methods, the only option is to retrain the model from scratch, which is costly and feasible only if the original data is still available. This highlights the importance of developing efficient methods for data removal while preserving model performance and minimizing costs.

This paper introduces a method for unlearning features and labels from a learning model. Inspired by *influence functions*, a technique from robust statistics [Hampel 1974], we develop a versatile approach that estimates the influence of data on learning models and translates it into a form of unlearning [Koh and Liang 2017, Koh et al. 2019]. This allows us to efficiently compute closed-form updates of model parameters, even for large datasets.

We demonstrate that our approach enables certified unlearning for models with strongly convex loss functions like logistic regression and support vector machines, providing theoretical guarantees on the removal of features and labels. However, for models with non-convex loss functions such as deep neural networks, similar guarantees cannot be achieved. Nevertheless, empirical results show that our approach is significantly faster compared to sharding and retraining methods while maintaining a similar level of accuracy.

We validate the effectiveness of our approach through case studies on unlearning label poisoning in computer vision.

Literature Survey

The increasing use of machine learning with personal data has sparked research into detecting and correcting privacy issues in learning models. Initially, retraining from scratch may seem like the best solution to address these issues. By directly correcting the training data, all potential issues can be reliably resolved. However, retraining from scratch presents several disadvantages:

- Retraining from scratch can be costly depending on the size of the original data. The entire learning process must be reproduced even if only a few data instances, features, or labels need correction.
- Privacy regulations require clear definition and approval of the purpose and duration of data storage by the user. Therefore, keeping the original data indefinitely for retraining may not be feasible.
- In online learning systems like chatbots, training data is volatile and may not be fully available. Yet, inappropriate memorization must be promptly removed.

In response to this situation, various concepts for machine unlearning have been proposed in recent years. Numerous methodologies have been explored in this field.

Bourtoule et al. [Bourtoule et al. 2020] introduced SISA training, a framework that facilitates faster unlearning by

minimizing the impact of individual data points during training. This approach, particularly advantageous for stateful algorithms, reduces the temporal and computational overhead associated with unlearning tasks.

In another paper by Chundawat et al. [Chundawat et al. 2022], authors introduced the zero-shot technique, which unlearns without access to any original training samples. This method provides defense against potential data attacks.

Methodology

In many learning-based systems, data points are linked directly to individuals, such as in face recognition systems where each data point is a portrait photo of a person. However, privacy issues can arise at different levels of data granularity. For example, the leaked address of a celebrity on social media can affect features of hundreds of data points, and sensitive information like personal names or phone numbers may appear in multiple emails.

Instance-based unlearning is inefficient in these cases for two main reasons: the runtime advantage over retraining diminishes as the number of affected data points increases, and removing entire instances degrades model performance unnecessarily.

To address this, we focused on unlearning individual features and labels. This strategy operates along an orthogonal dimension of the data. Instead of correcting privacy issues within data instances (columns), we focus on resolving them in feature values and labels (rows). This is feasible because we formulate unlearning as a closed-form update of the model, enabling us to correct features and labels at arbitrary positions in the training data.

Our method is built upon a supervised learning task defined by a dataset $D = \{z_1, \ldots, z_n\}$, where each $z_i = (x, y)$ represents features $x \in X$ and a label $y \in Y$. We assume $X = R^d$, and denote the j-th feature of x as $x[j]$. Given a loss function $l(z, \theta)$ that evaluates the difference between the predictions of a learning model θ and the true labels, the optimal model θ^* is obtained by minimizing the regularized empirical risk:

$$\theta^* = \arg\min_{\theta} L_b(\theta; D)$$

$$= \arg\min_{\theta} \sum_{i=1}^{n} l(z_i, \theta) + \lambda\Omega(\theta) + b^T\theta \tag{1}$$

Here, $L_b(\theta; D)$ represents the regularized empirical risk, λ is the regularization parameter, $\Omega(\theta)$ is the regularization term, and $b^T\theta$ is a bias term [Duda, Hart, and Stork 2000].

How would the optimal model θ^* change if only one data point z had been perturbed by some change δ? Replacing z by $\tilde{z} = (x + \delta, y)$ leads to the new optimal model. However, calculating the new model $\theta^*_{z \to \tilde{z}}$ exactly is expensive and does not provide any advantage over solving the problem in Equation 1. Instead of replacing the data point z with $\tilde{z}$, we can also up-weight $\tilde{z}$ by a small value ϵ and down-weight z accordingly, resulting in the following problem:

$$\theta^* = \arg\min_{\theta} L(\theta; D) + \epsilon l(\tilde{z}, \theta) - \epsilon l(z, \theta) \tag{2}$$

As a result, we do not need to explicitly remove a data point from the training data but can revert its influence on the learning model through a combination of up-weighting and down-weighting.

Equipped with a general method for updating a learning model, we proceed to introduce our approach for unlearning features and labels. To this end, we expand our notion of perturbations and include changes to labels by defining

$$\tilde{z} = (x + \delta_x, y + \delta_y),$$

where δ_x modifies the features of a data point and δ_y its label. By specifying different perturbations $\tilde{Z}$, we can now realize several unlearning tasks with closed-form updates.

Methods of Unlearning

Our approach rests on changing the influence of training data in a closed-form update. In the following, we derive two strategies for calculating this closed form: a **first-order update** and a **second-order update**. *The first order update* builds on the gradient of the loss function and thus can be applied to any model with differentiable loss. The second strategy incorporates second-order derivatives, which limits the application to loss functions with an invertible Hessian matrix.

We aim to find an update $\Delta(Z, \tilde{Z})$ that we can add to our model θ^* for unlearning. If the loss l is differentiable, we can compute an optimal first-order update as follows:

$$\Delta(Z, \tilde{Z}) = -\tau \left(\sum_{\tilde{z} \in \tilde{Z}} \nabla_\theta l(\tilde{z}, \theta^*) - \sum_{z \in Z} \nabla_\theta l(z, \theta^*) \right) \tag{3}$$

where τ is a small constant that we refer to as the unlearning rate. This update strategy is similar to a gradient descent update GD given by:

$$GD(\tilde{Z}) = -\tau \sum_{\tilde{z} \in \tilde{Z}} \nabla_\theta l(\tilde{z}, \theta). \tag{4}$$

However, it differs from this update step in that it adjusts the model based on the difference in gradient between the original and perturbed data, minimizing the loss on $\tilde{z}$ while removing the information contained in z. The first-order update is a straightforward yet effective strategy.

Second Order Update: The unlearning rate τ can be eliminated if we make further assumptions on the properties of the loss l. If we assume that l is twice differentiable and strictly convex, the second-order update is the preferred strategy for unlearning on models with a strongly convex and twice differentiable loss function that guarantees the existence of $H^{-1}_{\theta^*}$.

In contrast to the first-order update, however, this computation involves the inverse Hessian matrix, which can be difficult to construct for large learning models.

$$\Delta(Z, \tilde{Z}) = H^{-1}_{\theta^*} \left(\sum_{\tilde{z} \in \tilde{Z}} \nabla_\theta l(\tilde{z}, \theta) - \sum_{z \in Z} \nabla_\theta l(z, \theta) \right) \tag{5}$$

Calculating inverse Hessian: The Hessian matrix is a square matrix of second-order partial derivatives of a scalar-valued function. In the context of optimization, it represents the curvature of the loss surface around the current parameter values. The inverse Hessian matrix, provides information about the local geometry of the loss surface and is used to guide the optimization process. To calculate the inverse Hessian, given a model $\theta \in R^p$ with p parameters, forming and inverting the Hessian requires $O(np^2 + p^3)$ time and $O(p^2)$ space [Koh and Liang 2017]. For models with a small number of parameters, the matrix can be pre-computed and explicitly stored, so each subsequent request for unlearning only involves a simple matrix-vector multiplication.

Unlearning on Computer Vision

We concentrate on addressing a poisoning attack in computer vision. We simulate label poisoning, where an adversary partially flips labels between classes in the training data. While this attack does not affect privacy, it poses a security threat without altering features, making it an interesting scenario for unlearning. For this experiment,

- **Initial Model Training:** We utilize the CIFAR10 dataset [Krizhevsky 2009] and train a convolutional neural network with 1.8 million parameters, comprising three VGG blocks and two dense layers. The network achieves a reasonable performance of 87% accuracy without poisoning.
- **Poisoning Attack:** In poisoning labels, we identify pairs of classes and flip a fraction of their labels to their respective counterparts, for instance, from "cat" to "truck" and vice versa. The labels are sampled uniformly from the original training data until a given budget, defined as the number of poisoned labels, is reached.
- **Unlearning Task:** Our objective is to correct the flipped labels resulting from the poisoning attack. Specifically, we employ various first order and second order update unlearning methods with randomly selected labels for the attack.

Dataset: The CIFAR-10 dataset [Krizhevsky 2009] is a widely used benchmark dataset in machine learning and computer vision. It consists of 60,000 32x32 color images in 10 classes, with 6,000 images per class. The dataset is split into 50,000 training images and 10,000 test images. Each image is labeled with one of the following categories: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, or truck.

Model: We used a convolutional neural network (CNN) as our reference model, which consists of approximately 1.8 million parameters. This CNN architecture includes three VGG blocks followed by two dense layers. Each VGG block comprises convolutional layers with 128 filters of size 3×3, followed by max-pooling layers with a pooling size of 2×2. The rectified linear unit (ReLU) activation function is applied after each convolutional layer.

Hyperparameter Tuning: We train it using the Adam optimizer [Kingma and Ba 2015] with a learning rate of 1×10^{-4}. For our approach, we set an unlearning rate (τ) of 2×10^{-5} for the first-order update with batches of size 512. We use a batch size of 1,024 for the approximation of the inverse Hessian, with damping of 1×10^{-4} and scale of 2×10^5. The batch size determines the number of unlearning requests performed simultaneously, while the HVP batch size denotes the number of training samples used in the computation of the inverse Hessian.

Agarwal et al. [Agarwal, Bullins, and Hazan 2017] originally suggest repeating the algorithm multiple times and averaging the results for a more stable solution, but we find that a single repetition is sufficient in our experiments. Additionally, we introduce a patience parameter of 20 for the norm of the Hessian vector product, allowing for early stopping when the update norm does not decrease for a certain number of iterations.

Results

The initial model training by VGG achieved an accuracy of 87.86%. Table 1 shows detailed results before poisoning, while Figure 1 presents a graphical representation of the precision, recall, and F1-score achieved for each class.

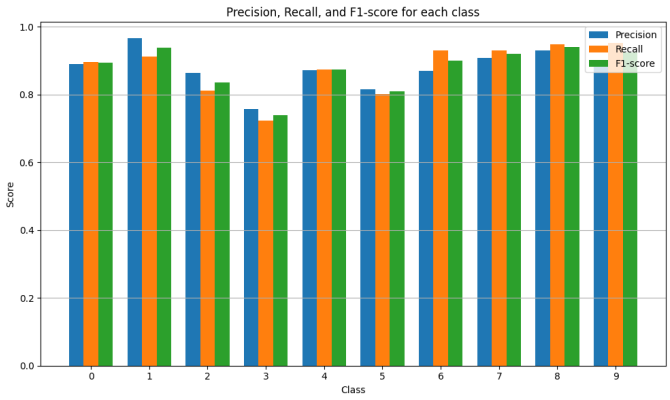

Figure 1: Bar Plot displaying precision, recall, and F1-score achieved for each class.

Metric	Value
Accuracy	0.8786
Train Loss	0.0041
Test Loss	0.6893
Weighted Avg	
Precision	0.8781
Recall	0.8786
F1-score	0.8779

Table 1: Initial Model Training Results on VGG19

Table 2 presents the final results achieved by this project, employing both the first-order and second-order update methods as discussed earlier. Figure 2 shows the confusion matrix displaying the class-wise classification results before unlearning and after the final unlearning process.

Metric	First Order	Second Order
Accuracy (Clean)	0.8786	0.8786
Accuracy (Before Fix)	0.7382	0.7382
Accuracy (After Fix)	0.7686	0.7830
Accuracy Restored (%)	21.65%	31.91%
Number of Gradients	512	3420
Unlearning Duration (s)	3327.04	8428.70
Number of Parameters	1798282	1798282

Table 2: Unlearning Results

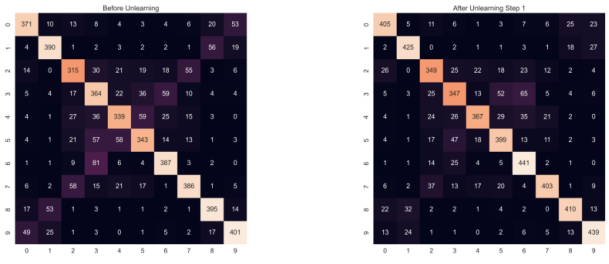

Figure 2: Confusion matrix displaying class-wise classification result before unlearning (left image) and after unlearning (right image)

Limitations

While our approach effectively removes features and labels in various experiments, it does have limitations that must be taken into account in practical applications.

Limits of Unlearning: The effectiveness of unlearning diminishes as the number of affected features and labels increases. While our approach can manage privacy leaks with hundreds of sensitive features and thousands of labels, it may struggle when dealing with changes affecting millions of data points. If our approach could correct changes of arbitrary size, it could potentially serve as a universal replacement for all learning algorithms — but that is clearly impossible [Wolpert and Macready 1997].

Unlearning Requires Detection: Lastly, it's important to note that our method relies on knowing which data to remove. Detecting privacy leaks in learning models is a challenging task beyond the scope of this work. The nature of privacy leaks varies depending on the data, learning models, and application.

Conclusion

We illustrate the effectiveness of our approach using the CIFAR-10 dataset. Our project successfully implemented unlearning on this image dataset, preserving the model's accuracy after a poisoning attack. Although the unlearning model restored the accuracy to 78.30%, compared to the accuracy before the attack (87.86%), further refinement is needed for near-accurate scores.

The unlearning process was applied to a computer vision dataset, but it can be adapted to work on a wide range of applications. For example, in the problem of unintended memorization, language models based on recurrent neural networks can accidentally memorize sensitive data, such as credit card numbers or private messages. Through specifically crafted inputs, an attacker can extract this data during text completion [Carlini et al. 2019].

References

Agarwal, N.; Bullins, B.; and Hazan, E. 2017. Second-order stochastic optimization for machine learning in linear time. *Journal of Machine Learning Research (JMLR)*, 4148–4187.

Bourtoule, L.; Chandrasekaran, V.; Choquette-Choo, C. A.; Jia, H.; Travers, A.; Zhang, B.; Lie, D.; and Papernot, N. 2020. Machine Unlearning. arXiv:1912.03817.

Carlini, N.; Liu, C.; Erlingsson, Ú.; Kos, J.; and Song, D. 2019. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *USENIX Security Symposium*, 267–284.

Chundawat, V. S.; Tarun, A. K.; Mandal, M.; and Kankanhalli, M. S. 2022. Zero-Shot Machine Unlearning. *CoRR*, abs/2201.05629.

Duda, R. O.; Hart, P. E.; and Stork, D. G. 2000. *Pattern Classification*. John Wiley & Sons, 2nd edition.

Hampel, F. 1974. The influence curve and its role in robust estimation. *Journal of the American Statistical Association*.

Kingma, D. P.; and Ba, J. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Koh, P. W.; Ang, K.; Teo, H. H. K.; and Liang, P. 2019. On the accuracy of influence functions for measuring group effects. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Koh, P. W.; and Liang, P. 2017. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning (ICML)*, 1885–1894.

Krizhevsky, A. 2009. Learning multiple layers of features from tiny images. University of Toronto, Tech. Rep.

Parliament, E.; and Council. 2016. Regulation 2016/679 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data. *Official Journal of the European Union*, 119: 1–88.

Wolpert, D. H.; and Macready, W. G. 1997. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(67).