Can Federated Models be Rectified through Learning Negative Gradients?

Ahsen Tahir, Zhiyuan Tan ✉, and Kehinde O. Babaagba

School of Computing, Engineering & the Built Environment, Edinburgh Napier University, Edinburgh EH10 5DT, United Kingdom {A.Tahir, Z.Tan, K.Babaagba,}@napier.ac.uk

Abstract. Federated Learning (FL) is a method to train machine learning (ML) models in a decentralised manner, while preserving the privacy of data from multiple clients. However, FL is vulnerable to malicious attacks, such as poisoning attacks, and is challenged by the GDPR's "right to be forgotten". This paper introduces a negative gradient-based machine learning technique to address these issues. Experiments on the MNIST dataset show that subtracting local model parameters can remove the influence of the respective training data on the global model and consequently "unlearn" the model in the FL paradigm. Although the performance of the resulting global model decreases, the proposed technique maintains the validation accuracy of the model above 90%. This impact on performance is acceptable for an FL model. It is important to note that the experimental work carried out demonstrates that in application areas where data deletion in ML is a necessity, this approach represents a significant advancement in the development of secure and robust FL systems.

Keywords: Federated Learning; Machine Unlearning; Negative Gradients; Model Rectification.

1 Introduction

The use of traditional Artificial Intelligence (AI) systems requires large amounts of data to be collected and stored on a central server or cloud storage. However, this can be difficult due to the various obstacles that can arise when transferring, collecting, and integrating data. In reality, data is often collected from edge devices such as smartphones and exists in isolated pieces. For example, health records are typically kept in separate, disconnected entities that are unable or unwilling to share data due to the privacy and security risks associated with sharing personal information.

Federated Learning (FL) introduced by Google [18,19,27], allows the training of Machine Learning (ML) models locally and the sharing of computations or model parameters to build a federated global model. FL enables a large number of participants to develop joint ML models without sharing their data and sacrificing their data privacy. However, recent work in FL revealed that poisoning

attacks [1,2,11] significantly temper the security of the federated global model. New data regulations, such as the General Data Protection Regulation (GDPR) and the California Consumer Privacy Act (CCPA), cast further challenges on data transactions. These new regulations emphasise on the user's right to be forgotten in light of current data regulations [14]. They raise the requirements for data deletion to restore the poisoned model or delete user data due to privacy regulations [3,10,15,16,21]. Although the problem can be solved trivially by retraining the model with the absence of the designated data from scratch, model retraining is not always sustainable and computationally feasible, especially in the current popularity of large models.

This leads to the emergence of machine unlearning, a set of strategies to eliminate the impact of training data on an ML model and its parameters in particular. Training data may need to be removed in order to fix a model that has been deliberately corrupted by a malicious user, or may need to be taken away by a user for privacy purposes. Removing this data and its effects from the ML model does not significantly reduce performance and is more computationally efficient than retraining the model from scratch. The ideal outcome is that the unlearned model is the same as or similar to the re-trained model.

In this paper, we introduce a machine learning technique based on negative gradients to reduce the effect of the selected subset of training data on a federated global model. We use the Federated Averaging (FedAvg) algorithm [27] to construct the global model in the context of this study. Additionally, we seek to answer two research questions.

1) To what extent does subtracting local model parameters serve to reduce the influence of the associated training data on the model to facilitate unlearning?
2) What impact does federated unlearning have on the performance of models?

The remainder of the paper is structured as follows. Section 2 provides a review of related work. Section 3 describes the methodology of our research. Section 4 presents and discusses the experimental results and analysis. Section 5 draws the conclusions of the investigation and the direction for future research.

2 Related Work

Federated models are created by combining updates from all participant models. The model aggregator carries out secure aggregation and does not have access to the updates given by individual models or their training data. The aggregation process makes federated models vulnerable to model poisoning attacks. Machine unlearning can be used when the model is contaminated/poisoned or a data deletion request is made by a client due to their right to be forgotten under current privacy laws.

2.1 Model Poisoning and Defending Mechanisms

In order to comprehend the process of unlearning in FL systems, it is essential to recognise the origins of model contamination. Model poisoning attacks can be classified into label-flipping attacks and backdoor attacks. Model poisoning incorporates data poisoning. Fig. 1 portrays the poisoning attack comprising of the poisoner objective, the genuine objective, and the element space which incorporates the attack target.

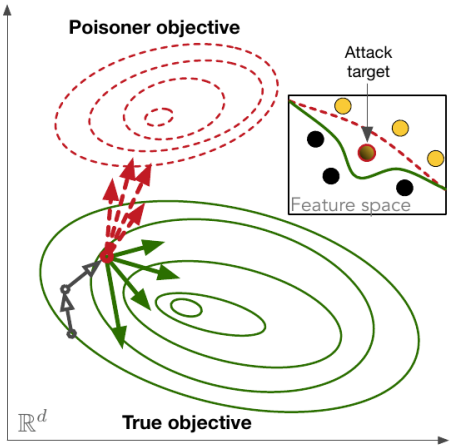


Fig. 1. Poisoning attack in Stochastic Gradient Descent (SGD). The red dotted vectors represent contributions from malicious clients (*Fung et al. [11]*)

Recent literature has explored poisoning attacks against FL. Sybil-based poisoning attack is studied in [11], which illustrates that a malicious client can increase the effectiveness of attacks by exploiting Sybils [9] and presents a defence strategy based on contribution similarity. The authors believed that a group of Sybils could potentially provide updates with a similar objective resulting in a lack of diversity amongst the malicious client updates.

The research, documented in [2], demonstrates a single malicious client-based model poisoning attacks against FL, where the updates from the malicious client were boosted to overwhelm the effect of benign clients. Furthermore, a strategy was introduced to keep malicious updates undetected by alternately optimising for the attack objective and training loss. The strategy keeps the validation accuracy of the resulting global model above a specified threshold and the pairwise distance between updates below a specific value to avoid attack detection.

In addition, backdoor attacks with a replacement model for FL are explored in [1]. The work proposed a constraint and scale technique for model replacement attacks, which uses the attacker's loss function during training to stay undetected.

2.2 Detection of Poisoned Models

Detecting a poisoned model in a federated setting is essential and has received increasing research interest [6, 7, 13, 23, 24, 32, 33]. Wang et al. [33] proposed the 'Neural Cleanse' model to identify and detect backdoor attacks in deep neural networks. An optimisation scheme is used to determine the minimal trigger required to cause misclassification of all samples from other labels into the target label. The size of each trigger is measured by the number of pixels it contains and an outlier detection algorithm is applied to identify any significantly smaller triggers, which represent real triggers for the backdoor attack.

In a similar fashion, [6] presents DeepInspect, a pioneering black-box backdoor detection and mitigation technique, which does not assume benign samples or require full access to the model being tested, unlike Neural Cleanse. Instead, it uses model inversion to create a substitution training set and a conditional Generative Adversarial Network (cGAN) to recover potential triggers used by the adversary for each output class. [24] proposes Artificial Brain Stimulation (ABS), a model-level backdoor detection method.

2.3 Formulation of Machine Unlearning

Machine unlearning is a process that focusses on the removal of certain subsets of training data to make it more efficient than the traditional retraining approach. This is especially useful when adapting an ML model to a changing environment, where a data subject may choose to exercise their right to be forgotten. By using this method, information from the data subset can be effectively removed from the trained model [36].

Recent studies are attempting to develop algorithms that can "exactly" or "approximately" remove the details of a particular subset of training data from a given ML model. In the context of machine unlearning, "Deletion" and "Efficiently Deletable" can be mathematically described as the following problems.

Definition of Deletion Let D be a dataset and A be a learning algorithm, such that

$$D \in \mathbb{R}^{n \times d} \tag{1}$$

$$A : D \rightarrow A(D) \in \mathcal{H} \tag{2}$$

where $\mathcal{H}$ refers to a hypothesis space (i.e., a model space or classifier space), such as linear regression models, and the function $A(D)$ returns model parameters or weights. If a data point i is deleted from the dataset D, then the remaining dataset can be represented as D_{-i}. Consequently, a model trained on the dataset D_{-i} can be represented as $A(D_{-i})$. The deletion operation R is another mapping in the hypothesis space.

$$R : (D, A(D), i) \rightarrow \mathcal{H} \tag{3}$$

$$\text{s. t. } R(D, A(D), i) = A(D - i) \tag{4}$$

where $R(D, A(D), i)$ returns a model that belongs to the space $\mathcal{H}$, defined by Eq. (3). The deletion operation, $R(D, A(D), i)$, defined in Eq. (4), results in a new model that is identical to that trained using the dataset D_{-i}. Eq. (4) implies that the learning algorithm is deterministic. If the learning algorithm is randomised, then Eq. (4) is then redefined as Eq. (5), which means that the two resulting models are equal in distribution.

$$R(D, A(D), i) \stackrel{d}{=} A(D_{-i}) \tag{5}$$

Definition of Efficiently Deletable In this work, we define *Efficiently Deletable* for the sequential learning algorithm A, which runs in time $\Omega(n)$. Ideally, the deletion operation, $R(D, A(D), i)$, is independent of the size of the dataset or, in other words, constant in time and is part of efficiently deletable. However, for the worst case, the algorithm is sublinear in time, which implies that an asymptotic lower bound for n data points and m deletion operations should be $\Omega(\frac{n}{m})$.

2.4 Unlearning Federated Learning

Studies have sought to create a generic unlearning process for various learning algorithms and ML models. Bourtoule et al. [3] proposed the well-known model-agnostic SISA framework, which divides the data into shards and slices. Each shard has a single model, and the final result is a combination of the various models in these shards. A model checkpoint is saved for each slice of a shard during training so that a new model can be re-trained from the interim state [3].

It may be difficult to create a model-agnostic unlearning framework due to the complexity of ML algorithms and the training process in the federated setting, where global weights are calculated using aggregation rather than simple gradients and multiple clients are involved [12]. Therefore, the early unlearning approaches cannot be simply transferred to the federated setting. In addition, clients may have some data that overlap, making it hard to determine the effect of each training item on model weights [25]. Applying conventional unlearning methods, such as gradient manipulation, can lead to a decrease in accuracy or extra privacy risks [20].

Data that need to be erased are assumed to belong to a single client in current research on federated unlearning [20, 25, 31, 34]. This assumption makes it easy to keep track of and delete the contributions a certain client has made to the global model training.

The gradients obtained from the client's data can be used to reverse the effects on the global model [21]. However, the unlearning hypothesis may not be the same as that initially learnt from the data. Removing the previous parameter adjustments could still damage the overall model. To address this issue, there are several potential solutions:

– Liu et al. [20] proposed calibration training to differentiate the individual contributions of clients as much as possible. However, this technique does not

work well for deep neural networks apart from basic architectures such as a 2-layer CNN or 2 fully connected layers. Additionally, there is a compromise between scalability and accuracy due to the cost of storing historical data on the federated server.
- Wu et al. [35] proposed a knowledge distillation strategy that uses a primary global model to train the unlearned model on the remaining data. However, since the server is unable to view the client's data, it is necessary to sample some unlabelled (fake) data that reflect the distribution of the entire dataset. This requires additional rounds of communication between the client and the server, which makes the process costly and approximate. Furthermore, if the data is non-IID [22] , the results may be further distorted.
- In a different area of the field, Liu et al. developed a smart retraining approach for federated unlearning without communication protocols [22] . This approach approximated the L-BFGS technique by using historical parameter changes to retrain the entire model. Unfortunately, this strategy is only suitable for models with fewer than 10,000 parameters and requires the storage of previous model snapshots, which contain past gradients and parameters that could potentially breach privacy.

2.5 Challenges to Federated Unlearning

Many ML algorithms are stochastic, which means they incorporate randomness in their optimisation or learning processes. This stochasticity is a fundamental concept in ML and must be understood to accurately interpret the behaviour of numerous predictive models. However, the randomness of ML algorithms presents additional difficulties for federated unlearning [21]. To make the unlearning hypothesis valid both mathematically and practically, the model to be unlearnt must start from the same state or with the same random weights as the original model. Additionally, randomness during optimisation could lead to different hypotheses that may correspond to different local minima in the optimisation space.

Furthermore, the probability of violating privacy and security increases with unlearning. This can take the form of:

- Information leakage [28]: The potential for unintentionally possessing confidential information can arise if a model is created using a mixture of confidential and non-confidential data.
- Reconstructing training data [5]: Malicious parties attempt to reconstruct the data used to train a model.
- Model inversion attacks [37]: A well-known privacy violation attempts to infer sensitive information from the training dataset, which leads to serious privacy issues.

Other serious implications can arise from adversarial exploitation [29], and systemic bias [17] among others. These can be addressed using a multifaceted approach that includes federated learning.

3 Methodology

A rudimentary approach to Federated Unlearning could involve retraining the model without the client's data, however, this is highly inefficient. Efficiently deleting data from ML models is a computational problem. Furthermore, it is important to formulate whether the data are "exactly" or "approximately" deletable.

To answer the questions raised in Section 1, we have introduced a technique that uses a negative gradient approach to unlearning the model. This method can be implemented on the client side without the need for sharing the client's data. The gradients obtained from the model trained on the client's data locally are used to subtract from the main global model. We reduce the impact of stochasticity by beginning with the same initial parameters and eliminating randomness in the training process.

The implementation of the proposed machine-unlearning solution involves the preparation of non-IID data for FL and the subsequent unlearning process. To achieve this, the unlearning hypothesis should be able to replicate the user data in a manner similar to how it is represented in the original model. This is because multiple hypotheses that follow different learning paths can be derived from the same data.

3.1 Experimental Data

Non-IID input data are generated from MNIST data [8], a large database of handwritten digits regularly used to train a variety of image processing systems systems, for Federated Learning (FL). Pre-processing of MNIST is performed using Leaf* [4], so that the data can be keyed by the original writer of the MNIST digits, because of the unique style of each writer, this dataset becomes the kind of non-IID data required for federated datasets.

The original 28×28 sized images are flattened into 784-element 1-D arrays, which are shuffled and organised in batches. The features are renamed from pixels and labelled input x and output y for use with PyTorch. The output y is multiclass with 10 output classes.

3.2 Network Implementation

A 5-layer deep neural network model with 3 fully connected hidden Rectified Linear Unit (ReLU) layers is employed to train the MNIST images in PyTorch 1.7.1 [30] with torchvision 0.7.0 [26]. The Federated Averaging (FedAvg) algorithm is implemented in PyTorch to add gradients and merge the local user models into a federated main model. The FedAvg algorithm uses two optimisers: a client side and a server optimiser. The client optimiser computes updates to the local model, while the server optimiser averages the updates in the global model. The process is tested with regular Stochastic Gradient Descent (SGD).

The implemented model ran with the following parameters:

* https://github.com/TalwalkarLab/leaf

* Learning rate, $\eta = 0.01$.
* SGD learning algorithm, Momentum = 0.9.
* Number of Epochs, $\epsilon = 10$.
* Batch size = 32.
* The loss function, CrossEntropyLoss(), is applied.

Five local client models were used to build the federated main model. Each of the client models was trained on 10,000 samples, validated on 1,000, and was also tested on 1,000 samples.

4 Experimental Results and Discussion

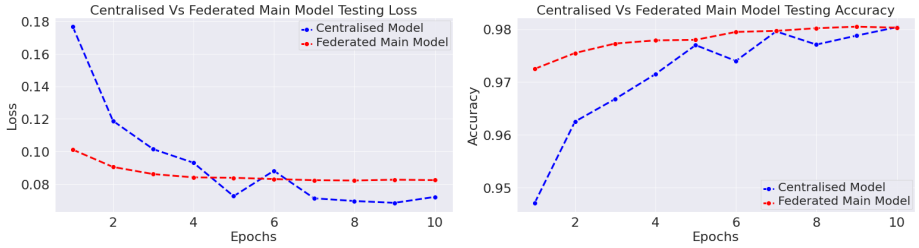
Two deep learning-based image recognition models are compared by training them (1) centrally on all data and (2) with FL which trains five local client models on their respective selected subsets of data, then merges them into a main model using the FedAvg algorithm. Experimental results are depicted in Figs. 2 and 3.

(a) The training and test loss rates of the centrally trained model.

(b) The training and test accuracy rates of the centrally trained model.

Fig. 2. Preliminary experimental results of the centrally trained model.

– The model that was trained centrally achieved the highest training accuracy of 99.54% and the highest test accuracy of 98.04%, as illustrated in Fig. 2(b). Further details of the cross-entropy loss function and the training/test accuracy with the number of epochs can be seen in Figs. 2(a) and 2(b).
– The federated main model performs remarkably well, attaining a test accuracy of 98. 01% across the entire dataset, as seen in Fig. 3(b). It should be noted that the federated main model begins with a much lower loss and a much higher accuracy than the main model did in its initial run, as demonstrated in Figs. 3(a) and 3(b).



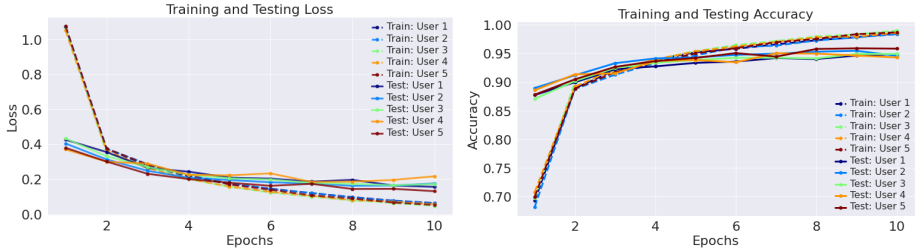
(a) The comparison of the test loss rates of the centrally trained model and the federated main model.

(b) The comparison of the test accuracy rates of the centrally trained model and the federated main model.

Fig. 3. The comparison of the preliminary experimental results of the centrally trained model and the federated main model.

4.1 Federated Main Model

In this section, we provide a comprehensive analysis of the federated main model, which is established by averaging the five local models that have been individually trained on their own subset of data. Figs. 4(a) and 4(b) illustrate the cross-entropy loss and accuracy of the model training and test in relation to the number of epochs for the five models. The curves for the loss and accuracy of model training and testing of all five models are very similar.



(a) The training and test loss rates of the five local client models.

(b) The training and test accuracy rates of the five local client models.

Fig. 4. Experimental results of the five local client models.

The detailed performance of the local client models on their respective partitioned datasets is given in Table 1. The local models of the five client/users provide good accuracy rates of 94.28%, 95.39%, 94.44%, 94.83% and 96.04%, respectively. The removal of each of the local models from the federated main model reduces the accuracy of the main model in the respective partitions, as illustrated in Table 1. The results show that

Table 1. Merged federated main model vs local models on their respective data before and after model removal

Dataset Partition	Local Client Model (%)	Federated Main Model (%)	Federated Main Model with Local Client Model 1 Removed (%)	Federated Main Model with Local Client Model 2 Removed (%)	Federated Main Model with Local Client Model 3 Removed (%)	Federated Main Model with Local Client Model 4 Removed (%)	Federated Main Model with Local Client Model 5 Removed (%)
1	0.9428	0.9567	0.9183	0.9306	0.9067	0.8639	0.7183
2	0.9539	0.9617	0.9261	0.9444	0.9156	0.8717	0.7267
3	0.9444	0.9572	0.9089	0.92	0.89	0.85	0.7228
4	0.9483	0.9589	0.9167	0.9239	0.9117	0.8678	0.7167
5	0.9604	0.9665	0.918	0.9291	0.909	0.8638	0.7166

- Removing any of the local model parameters from the main model will cause a decrease in the resulting federated main model's performance on all partitions of the dataset,
- The resulting federated main model may not show lower accuracy on the data that was removed from the original federated main model than on the other partitions of the dataset,
- No relationship exists between the accuracy of the local model and that of the resulting federated main model when the parameters of the local model are removed from the original federated main model,
- The accuracy of the federated main model is reduced from 96.17% to 94.44%, when the Local Client Model 2 with an accuracy of 95.39% is subtracted from the main model,
- The accuracy of the federated main model is reduced to 91.83% when Local Client Model 1 with an accuracy of 94.28% is subtracted from the main model.
- Subtraction of Local Client Model 2 with accuracy 95.39% produces the least reduction in the accuracy of the federated main model and subtraction of Local Client Model 5 with accuracy 96.04% results in the largest reduction in the accuracy of the performance of the federated main model.

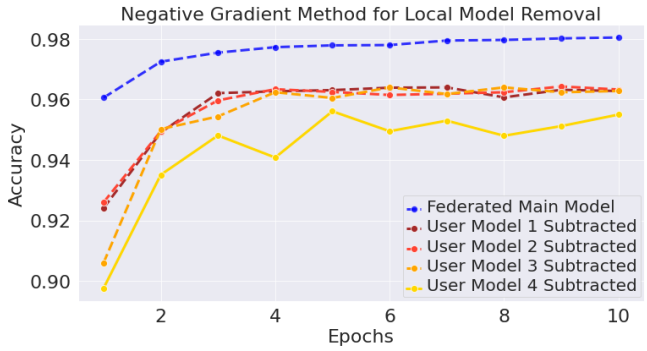


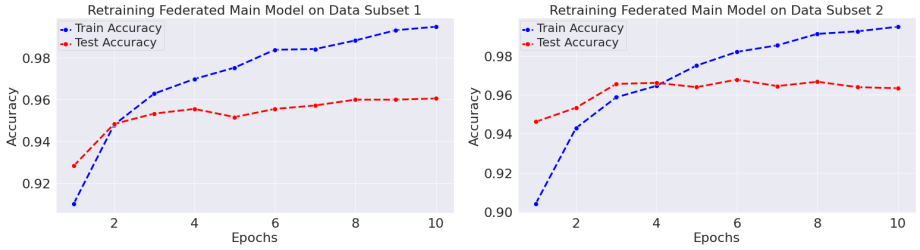
Fig. 5. The performance of the federated main model after client model subtraction

The implementation of the negative gradient method results in the subtraction of the parameters of the local client model from the federated main model at each epoch, illustrating a graph of the lower performance testing accuracy across the entire dataset in Fig. 5. The figure illustrates that different local models can potentially have different effects on the accuracy of the federated main model.

4.2 Retraining of Federated Main Model

Once the parameters of each local client model have been taken away from the original federated main model one at a time, the various resulting federated

main models are re-trained using the data from the models that were taken away directly. The FedAvg algorithm is no longer involved; instead, local retraining is used to update the federated model. This experiment demonstrates that after unlearning the federated main model can be used to learn the same or new data on the server or by a new user. Then, it can reach the same performance levels as before. The retraining of the federated main model was then carried out using data subsets 1 and 2. Fig. 6 shows the retraining of the federated main model with data subsets 1 and 2.



(a) Retraining of federated main model on data subset 1

(b) Retraining of federated main model on data subset 2

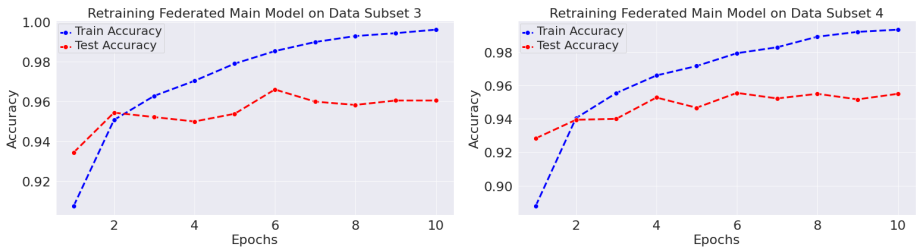
Fig. 6. Re-training of the federated main model after unlearning the contributions of Local Client Models 1 and 2 using the negative gradient technique.

Local Client Models 1 and 2 were trained on the data subsets 1 and 2 and then removed from the original federated main model using the negative gradient technique. Subsequently, the resulting federated main model was re-trained with the data subsets 1 and 2. The graph in Fig. 6(a) shows that the test accuracy of the federated learning model with the Local Client Model 1 subtracted (corresponding to the data subset 1) increased from 93. 44% to 96. 06% in 10 epochs. Fig. 6(b) shows the retraining of a federated main model with the Local Client Model 2 removed. This resulted in a 10-epoch improvement from 94.61% to 96.33%.

The retraining of the federated main model was then performed on the data subsets 3 and 4. The negative gradient method was used to remove Local Client Models 3 and 4 from the federated main model. Subsequently, the federated main model was re-trained using data subsets 3 and 4, as shown in Fig. 7. Figs. 7(a) and 7(b) show that the accuracy of the federated learning model with the Local Client Models 3 and 4 subtracted, respectively, increased from 92.5% and 92.83% to 95.94% and 95.50% respectively, over the course of 10 epochs.

5 Conclusion

Federated unlearning has been shown to be a viable solution to the issues caused by data deletion in Federated Learning (FL). This approach involves the removal



(a) Retraining of federated model on data subset 3

(b) Retraining of federated model on data subset 4

Fig. 7. Retraining of federated main model after removal of user models 3 and 4 through negative gradient technique.

of unwanted updates from a selection of clients, which can help reduce the negative impact of data deletion on the accuracy and dependability of federated learning models.

This paper has introduced a technique for machine unlearning by using negative gradients of local models to erase their effect on the main model based on FL. The results showed that subtracting local model parameters can be used to eliminate the influence of the corresponding training data on the model and unlearn it for the FL paradigm. Although there was a decrease in the model's performance on the data due to deletion, the performance of the model on the overall data remained above 90%. It is important to note that the experimental work conducted demonstrates that in application areas where data deletion in machine learning is necessary, such as when a client may need to remove their data from the model, potentially exercising the right to be forgotten, the model performance will not be significantly affected.

Despite the fact that there are still difficulties to be solved in the implementation of federated unlearning, such as the possibility of privacy infringements and the necessity of efficient detection of poisoning assaults, this method represents a major advance in the formation of secure and reliable federated learning systems.

Acknowledge

This work was supported by ENU Development Trust Ref. LH Oct19.

References

1. Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., Shmatikov, V.: How to backdoor federated learning. In: Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics, PMLR. vol. 108, pp. 2938–2948 (2020)
2. Bhagoji, A.N., Chakraborty, S., Seraphin Calo, P.M.: Analyzing federated learning through an adversarial lens. In: Proceedings of the 36th International Conference on Machine Learning, PMLR. vol. 97, pp. 634–643 (2019)

3. Bourtoule, L., Chandrasekaran, V., Choquette-Choo, C., Jia, H., Travers, A., Zhang, B., Lie, D., Papernot, N.: Machine unlearning. In: Proceedings of the 42nd IEEE Symposium on Security and Privacy (May 2021)
4. Caldas, S., Duddu, S.M.K., Wu, P., Li, T., Konečnỳ, J., McMahan, H.B., Smith, V., Talwalkar, A.: Leaf: A benchmark for federated settings. In: Workshop on Federated Learning for Data Privacy and Confidentiality (2019)
5. Chai, X., Tang, G., Wang, S., Peng, R., Chen, W., Li, J.: Deep learning for regularly missing data reconstruction. IEEE Transactions on Geoscience and Remote Sensing **58**(6), 4406–4423 (2020). https://doi.org/10.1109/TGRS.2020.2963928
6. Chen, H., Fu, C., Zhao, J., Koushanfar, F.: Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19. pp. 4658–4664. International Joint Conferences on Artificial Intelligence Organization (7 2019). https://doi.org/10.24963/ijcai.2019/647, https://doi.org/10.24963/ijcai.2019/647
7. Chen, H., Fu, C., Zhao, J., Koushanfar, F.: Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks. In: IJCAI. pp. 4658–4664 (2019)
8. Deng, L.: The mnist database of handwritten digit images for machine learning research [best of the web]. IEEE Signal Processing Magazine **29**(6), 141–142 (2012)
9. Douceur, J.R.: The sybil attack. In: International workshop on peer-to-peer systems. pp. 251–260. Springer (2002)
10. Fu, S., He, F., Xu, Y., Tao, D.: Bayesian inference forgetting. arXiv preprint arXiv:2101.06417 (2021)
11. Fung, C., Yoon, C.J., Beschastnikh, I.: Mitigating sybils in federated learning poisoning. arXiv **abs/1808.04866** (2018)
12. Gao, X., Ma, X., Wang, J., Sun, Y., Li, B., Ji, S., Cheng, P., Chen, J.: Verifi: Towards verifiable federated unlearning. arXiv preprint arXiv:2205.12709 (2022)
13. Gao, Y., Xu, C., Wang, D., Chen, S., Ranasinghe, D.C., Nepal, S.: Strip: A defence against trojan attacks on deep neural networks. In: Proceedings of the 35th Annual Computer Security Applications Conference. pp. 113–125 (2019)
14. Garg, S., Goldwasser, S., Vasudevan, P.N.: Formalizing data deletion in the context of the right to be forgotten. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 373–402. Springer (2020)
15. Ginart, A., Guan, M.Y., Valiant, G., Zou, J.: Making ai forget you: Data deletion in machine learning. arXiv preprint arXiv:1907.05012 (2019)
16. Golatkar, A., Achille, A., Soatto, S.: Eternal sunshine of the spotless net: Selective forgetting in deep networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9304–9312 (2020)
17. Huang, X., Zhu, D., Zhang, F., Liu, T., Li, X., Zou, L.: Sensing population distribution from satellite imagery via deep learning:model selection, neighboring effects, and systematic biases. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing **14**, 5137–5151 (2021). https://doi.org/10.1109/JSTARS.2021.3076630
18. Konečnỳ, J., McMahan, H.B., Ramage, D., Richtárik, P.: Federated optimization: Distributed machine learning for on-device intelligence. arXiv preprint arXiv:1610.02527 (2016)
19. Konečnỳ, J., McMahan, H.B., Yu, F.X., Richtárik, P., Suresh, A.T., Bacon, D.: Federated learning: Strategies for improving communication efficiency. arXiv preprint arXiv:1610.05492 (2016)

20. Liu, G., Ma, X., Yang, Y., Wang, C., Liu, J.: Federaser: Enabling efficient client-level data removal from federated learning models. In: 2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS). pp. 1–10 (2021). https://doi.org/10.1109/IWQOS52092.2021.9521274
21. Liu, Y., Ma, Z., Liu, X., Ma, J.: Learn to forget: User-level memorization elimination in federated learning. arXiv preprint arXiv:2003.10933 (2020)
22. Liu, Y., Xu, L., Yuan, X., Wang, C., Li, B.: The right to be forgotten in federated learning: An efficient realization with rapid retraining. In: IEEE INFOCOM 2022 - IEEE Conference on Computer Communications. pp. 1749–1758 (2022). https://doi.org/10.1109/INFOCOM48880.2022.9796721
23. Liu, Y., Lee, W.C., Tao, G., Ma, S., Aafer, Y., Zhang, X.: Abs: Scanning neural networks for back-doors by artificial brain stimulation. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. pp. 1265–1282 (2019)
24. Liu, Y., Lee, W.C., Tao, G., Ma, S., Aafer, Y., Zhang, X.: Abs: Scanning neural networks for back-doors by artificial brain stimulation. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. p. 1265–1282. CCS '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3319535.3363216, https://doi.org/10.1145/3319535.3363216
25. Ma, Z., Liu, Y., Liu, X., Liu, J., Ma, J., Ren, K.: Learn to forget: Machine unlearning via neuron masking. IEEE Transactions on Dependable and Secure Computing **20**(4), 3194–3207 (2023). https://doi.org/10.1109/TDSC.2022.3194884
26. Marcel, S., Rodriguez, Y.: Torchvision the machine-vision package of torch. In: Proceedings of the 18th ACM international conference on Multimedia. pp. 1485–1488 (2010)
27. McMahan, H.B., Moore, E., Ramage, D., y Arcas, B.A.: Federated learning of deep networks using model averaging. arXiv preprint arXiv:1602.05629 (2016)
28. Milburn, A., Van Der Kouwe, E., Giuffrida, C.: Mitigating information leakage vulnerabilities with type-based data isolation. In: 2022 IEEE Symposium on Security and Privacy (SP). pp. 1049–1065 (2022). https://doi.org/10.1109/SP46214.2022.9833675
29. Oosthoek, K., Doerr, C.: Cyber security threats to bitcoin exchanges: Adversary exploitation and laundering techniques. IEEE Transactions on Network and Service Management **18**(2), 1616–1628 (2021). https://doi.org/10.1109/TNSM.2020.3046145
30. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. In: 33rd Conference on Neural Information Processing Systems (NeurIPS2019) (2019)
31. Thudi, A., Deza, G., Chandrasekaran, V., Papernot, N.: Unrolling sgd: Understanding factors influencing machine unlearning. In: 2022 IEEE 7th European Symposium on Security and Privacy (EuroSP). pp. 303–319 (2022). https://doi.org/10.1109/EuroSP53844.2022.00027
32. Veldanda, A.K., Liu, K., Tan, B., Krishnamurthy, P., Khorrami, F., Karri, R., Dolan-Gavitt, B., Garg, S.: Nnoculation: Broad spectrum and targeted treatment of backdoored dnns. arXiv preprint arXiv:2002.08313 (2020)
33. Wang, B., Yao, Y., Shan, S., Li, H., Viswanath, B., Zheng, H., Zhao, B.Y.: Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In: 2019 IEEE Symposium on Security and Privacy (SP). pp. 707–723. IEEE (2019)

34. Wang, J., Guo, S., Xie, X., Qi, H.: Federated unlearning via class-discriminative pruning. In: Proceedings of the ACM Web Conference 2022. p. 622–632. WWW '22, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3485447.3512222, https://doi.org/10.1145/3485447.3512222
35. Wu, C., Zhu, S., Mitra, P.: Federated unlearning with knowledge distillation. arXiv preprint arXiv:2201.09441 (2022)
36. Xu, H., Zhu*, T., Zhang, L., Zhou, W., Yu, P.S.: Machine unlearning: A survey. ACM Comput. Surv. (jun 2023). https://doi.org/10.1145/3603620, https://doi.org/10.1145/3603620, just Accepted
37. Zhang, Z., Liu, Q., Huang, Z., Wang, H., Lee, C.K., Chen, E.: Model inversion attacks against graph neural networks. IEEE Transactions on Knowledge and Data Engineering pp. 1–13 (2022). https://doi.org/10.1109/TKDE.2022.3207915