Federated Unlearning with Projected Gradient Ascent for FeatureCloud

Kester Bagemihl, Simon Feldmann, Antonia Gocke, Caterina Roncalli

03.02.2023

Abstract

Federated learning (FL) has recently emerged as a promising tool for privacy preserving distributed learning. With the GDPR endorsing the "right to be forgotten" it has become imperative to implement reliable data unleanring alorithms. The privacy concern is especially prevalent in the medical field where the highly sensitive data can be easily exploited. A platform wanting to tackle this issue is the FeatureCloud, which uses FL amongst other privacy preserving techniques to ensure safe collaborations between different medical research centers.
Intrinsically FL has a data heterogenity problem, which originates from the different clients gathering different data. Thus we are proposing an unlearning algorithm that unlearns the global model at the client level by maximizing the loss. The problem is considered as a constrained maximization problem. The constraint is set by a reference model calculated from the global learned model. Afterwards the unlearned model is relearned with the data of the remaining clients, generating a functional model.
This method can be used as an add-on feature, without saving any parameter updates or needing access to the data from the other clients. Experiments on the *MNIST* dataset show that the algorithm is reliable and effective at unlearning data up to a moderate level of heterogenity.

1 Introduction

Over the past decade Machine Learning (ML) has started to play an important role in the medical domain. Typical applications of ML algorithms include diagnostics of diseases, prediction of outcome and recommendation of treatments. ML models can be a powerful tool and can achive an even more accurate performance than humans, but the success relies heavily on the availability and quality of the data that is used to train such models [1]. The dataset needs to be large and diverse in order to prevent overfitting and minimize insitutional biases, justifying the need for multi-institutional collaborations across different hospitals or other medical organizations [2].
When working with medical data privacy is a huge concern as it is rigorously restricted. In Europe the General Data Protection Regulation (GDPR) governs the storage and exchange of data concerning health, genetic data, biometric data and personally identifiable data. One of the most important aspects is the right to have personal data removed, which is know as the "right to be forgotten" [3].
In the field of secure and privacy preserving AI several techniques, such as differential privacy, homomorphic encryption or secure mulit-party computation have emerged in order to protect highly sensitive data while still making it utilizable for research purposes [4, 5, 6, 7, 8, 9]. To further enhance privacy and adress the concerns regarding transfer of data, the concept of Federated learning (FL) was developed. Federated learning is a distributed machine learning approach, that enables the training of a global model on decentralized data. The data is kept distributed among multiple contributing parties (clients). In a setup with N clients, each client has a dataset $D_i = \{(\mathrm{x}_i, y_i)_{i \in [n_i]}\}$ with $[n_i] = \{1, 2, ..., n_i\}$. During the training process each client refines the

parameters at a specific timestep w_i^t of its local model based on an optimization algorithm. The most widely used method is gradient descent, where for each iteration the gradient of the loss function ∇F_i is computed with respect to the current models parameters. The parameter update follows as $w_i^{t+1} = w_i^t - \eta \nabla F_i(w_i^t))$, where η is the learning rate. These parameter updates are sent to a central server, where they are aggregated, using federated averaging. Hereby a weighted average of the parameters is computed to obtain the parameters of the global model for the next training round: $w^{t+1} = \sum_{i=1}^{n} p_i w_i^t$, where $p_i = \frac{n_i}{\sum_{i=1}^{N} n_i}$. These global parameters are sent again to all the participating clients and the iterative process is repeated for a specific number of T_{tr} training rounds. While FL has been shown to be a useful tool for privacy preservence, it still faces performance challenges, when it comes to data that is not independent and identically distributed (IID) across the clients. Although data that is diverse and varied in nature, is necessary to train robust and accurate models by reducing biases, it might however prevent a FL model from converging to an optimum or take a much longer time to train. [10, 11]

In the last years multiple platforms emerged to support federated learning [12] [13], among these the FeatureCloud platform [14] was developed. It allows the usage of several preimplemented machine learning applications in a federated learning setting, combined with different privacy enhancing techniques and is especially targeted at scientists from the medical domain.
While FL or FL with a combination of the above mentioned privacy enhancing techniques are applied on a regular basis in order to protect sensitive information, they do not adress the "right to be forgotten", stated in the GDPR. Even though the data might be encrypted and can not be tracked back to an individual, it will not be removed from a trained ML model. This results in the necessity to develope unlearning algorithms in order to eliminate the contribution of some specific training data on a ML model.

1.1 Federated Unlearning

As federated unlearning has shown to prove a big challenge it has not yet been investigated as thoroughly as machine unlearning, however there are a few notable publications.
The first tool for federated unlearning published is the FedEraser [15]. The FedEraser unlearns the model by keeping the prameter updates during the learning process of the model and reconstructing the unlearned model according to the updates. Using the FedEraser one calculates an unlearned model four times faster than retraining a model without the target data. A further speed up can be achieved by using an algorithm developed by Wu et. al [16] which introduces the principle of knowledge distillation to only save the relevant parameter updates, thus saving on storage space and time needed for the reconstruction. This principle however additionally needs some unlabeled outsourced data, which causes addditional privacy concerns especially for medical data. Further challenges provided by these two algorithms [15] [16] are the space needed to save the parameter updates and that they need to be implemented before learning a model in a federated fashion, thus limiting its applicability. A different approach was used by Halimi et. al [17] where the last local model of the target client is unleaned with the target clients data using a projected gradient ascent. This unlearned model is then retrained with only a limited number of rounds of federated learning, achiving a high accuracy. In order to minimize the chance of creating a random model during the gradient ascent, the algorithm ensures, that the unlearned model is sufficiently close to a previously computed *reference model.* This reference model is calculated as the average of the remaining clients model parameter, i.e $w_{ref} = \frac{1}{N-1} \sum_{j \neq i} w_j^{T-1} = \frac{1}{N-1}(N w^T - w_i^{T-1})$. Bounded by a ℓ_2-norm ball of radius δ, the target client then optimizes the models parameters with the before mentioned projected gradient ascent. The ℓ_2-norm ball of radius δ around w_{ref} is denoted as $\Omega = \{v \in \mathbb{R}^d : ||v - w_{ref}||_2 \leq \delta\}$. Defining $\mathcal{P} : \mathbb{R}^d \to \mathbb{R}^d$ as the projection operator onto Ω, the parameter updates follow as $w \leftarrow \mathcal{P}(w_i + \eta_u \nabla F_i(w_i))$. The authors also include an additional early stopping criterium τ based on the validation accuracy on a previously split client validation set. After the unlearning process the unlearned model is returned to the server and T_{re} rounds of federated learning with the remaining clients are performed. In our study we select the already existing federated un-

learning approach by Halimi et. al [17] and aim to archive better results through modifications to the algorithm. Furthermore, we set a focus on the applicability of the unlearning algorithm to heterogeneously distributed data.

1.2 Backdoor Verification

An additional obstacle when tackling the unlearning problem is the verification of the unlearning process.
A method to certify the data removal was published by Guo et al. [18] where systematically poisoned data [19] labelled "backdoor data" is trained with the model.After the unlearning process the accuracy is determined on clean data and the backdoored data. With a low accuracy in the backdoor data and a high accuracy on the clean data, it can be concluded, that the model unlearned the backdoor data, whilst still being a functional model for the clean data.

2 Methods

Training Setup
Inspired by Halimi et. al we split up the training data among N clients and we perform FL for T_{tr} rounds to obtain a global model. Afterwards the contribution from one of the clients, the *target client*, is being removed by applying the unlearning algorithm. Finally we retrain the resulting unlearned model with the remaining N-1 clients for T_{re} rounds.
As described in section 1.1 the model to which the unlearning algorithm is applied, referred to as the *unlearning model*, is the local model of the target client from the last round of federated training. As our results show poor performance regarding the success of unlearning, we initialize the unlearning model as the global model from the current timestep and apply the projected gradient ascent. The pseudocode for this modified version can be found in Algorithm 1

Algorithm 1 Federated Unlearning via Projected Gradient Ascent

Unlearning Parameters
learning rate η_u, batch size B_u, number of epochs E_u, clipping radius δ, and early stopping threshold τ.

```
1:  Split dataset $D_i$ into trainset $D_i^{tr}$ and validation set $D_i^{val}$
2:  Set $w_{ref} \leftarrow \frac{1}{N-1}(Nw^T - w_i^{T-1}) = \frac{1}{n-1}\sum_{i\neq j} w_j^{T-1}$
3:  Define $\mathcal{P}(w)$ as the projection of $w \in \mathbb{R}^d$ onto the $l_2$-norm ball $\Omega = v \in \mathbb{R}^d : ||v - w_{ref}||_2 \leq \delta$
4:  Initialize unlearning model as $w \leftarrow w_{glob}$
5:  for each epoch $e \in E_u$ do
6:      for each batch $b \in D_i$ do
7:          $w \leftarrow \mathcal{P}(w + \eta_u \nabla F_i^u(w; b))$
8:          if Accuracy $(w; D_i^{val}) \leq \tau$ then
9:              Set $w_i^u \leftarrow w$
10:             $w_i^u$ to server
11:         end if
12:     end for
13: end for
14: Set $w_i^u \leftarrow w$
15: $w_i^u$ to server
```

Dataset and Data split
The study was conducted on the MNIST Dataset [20] consisting of (28×28) pixel images of handwritten digits with ten different classes. The dataset contains 60000 training images (of which 30% are used as validating images) and 10000 testing images. The class distribution in the original dataset is fairly balanced (Appendix A.3, Figure 1). For Fedeated Learning the

training data D is split equally among N = $\{3, 5, 10\}$ clients, such that the classes are distributed independent and identically (IID) across the clients. To analyze the influence of data heterogenity the data is furthermore distributed non-IID among the same number of clients, such that each client has a similar number of total training samples, but the class distribution is imbalanced. We define two levels of non-IID - mild and extreme - and divide the classes among the clients following that definition. Detailed information about the data split are listed in Table 1 and the distributions are visualized in Appendix A.3, Figures 2 - 7.

Distribution	**Dataset**	**N**	**k**
independent and identically distributed	IID$_3$	3	3
	IID$_5$	5	5
	IID$_{10}$	10	10
mildly non independent and identically distributed	nIID$_{3,2}$	3	2
	nIID$_{5,3}$	5	3
	nIID$_{10,5}$	10	5
strongly non independent and identically distributed	nIID$_{3,1}$	3	1
	nIID$_{5,1}$	5	1
	nIID$_{10,2}$	10	2

N: Number of clients among which the data is split
k: Number of clients each class is assigned to

Table 1: Class distribution for IID and non-IID data split among different numbers of clients. The notation of the Dataset is IID$_N$ for independent and identically distributed data and nIID$_{N,k}$ for non independent and identically distributed data.

Model Architecture and Training Details
We used a CNN with four convolutional layers with 5×5 filter size, followed by two fully connected layers. A detailed description of the models' architecture can be found in Appendix C, Table 12. Both in the centralized and federated set up we train the model with Adam [21] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, a learning rate of 0.001 and weight decay of 0.5.

Evaluation with backdoor triggers
To evaluate the performance of the unlearning algorithm we follow the approach of Halimi et al. [17] and use backdoor triggers. The data of the client who will be unlearned has a probability of p_{BD}, to be augmented with a backdoor trigger, while the data of the other clients remains clean. This makes the global model susceptible to these triggers, resulting in a high accuracy on a backdoored test set. If the unlearning process was successful, the unlearned global model should have a low accuracy on backdoored data, while still remaining a high accuracy on the MNIST test set. As a backdoor trigger we use a (3×3) pixel patch with maximum intensity value and add it to a fraction of the target clients data. The label of this backdoored data is changed to '9' and the data whose label is already the target label is excluded. To evaluate the backdoor attacks we augment the images in the MNIST test set with a probability of $p_{BD} = 1$ with a backdoor trigger and use it as our backdoor test set.

2.1 Experiments

2.1.1 Baseline Experiments

To verify that the federated learning itself is successful, we compare the accuracy on the MNIST test set achieved by training a model in a centralized fashion (**Cent-1.1**) with the accuracy achieved by training a model in a federated fashion (**Experiment Fed-1.1**). The training is

repeated for all nine client splits and executed until convergence. As we only use clean data for the training we not only evaluate the accuracy on MNIST test set, but also on the backdoor test set to set a baseline for a model, that has never seen backdoored data.
Furthermore to compare the efficiency of the unlearning algorithm to a 'retraining from scratch approach' without the target client, we train a model with only the data of the remaining N-1 clients in a federated fashion (**Experiment Fed-1.2**). The model is trained for one round and 20 rounds, to determine whether one round of federated learning after unlearning has a better performance than training from scratch. The training is repeated for all different client splits, but without the participation of the target client.

2.1.2 Experiments for Evaluation of Unlearning

The evaluation of the original unlearning approach, where the local client model is used as the unlearning model, is conducted on the IID, mild and strong non-IID data split for three clients (**Experiment Fed-2.1**). The target clients data has a probability of $p_{BD} = 0.3$ to be augmented with the backdoor trigger. The model is trained in a federated fashion, such that the accuracy on the MNIST test set and backdoor test set converge and are subsequently unlearned with the early stopping criterion $\tau = 0.12$ and δ being set to one third of the average Euclidean distance between w_{ref} and a random model, where the average is computed over 10 random models. After the unlearning the model is retrained for one and 20 rounds. As the results for the accuracy fluctuate, we repeat the training process with the same set up five times.
The evaluation of our modification to the unlearning algorithm, where we use the global model as the unlearning model, is conducted on all nine client splits (**Experiment Fed-2.2**). The target client has a probability of $p_{BD} = 0.3$ for N=3, $p_{BD} = 0.66$ for N=5 and $p_{BD} = 0.8$ for N=10 to be augmented with the backdoor trigger, analogously to Halimi et. al [17]. Apart from using a different model as the unlearing model, the training set up is the same as in Experiment 2.1.
A detailed description of the hyperparameters used in the individual experiments can be found in Appendix A.1.

3 Results

3.1 Results for Baseline Experiments

In Experiment Cent-1.1 and Fed-1.1 we compare the MNIST accuracy for centralized and federated learning with different client splits with clean data. The highest accuracy with 0.991 is achieved by training in a centralized fashion (Appendix A.2, Table 9). The MNIST accuracy achieved by federated training is comparibly high for the IID and mildly non-IID client splits and slightly lower for the strong non-IID client splits with the lowest accuracy of 0.995 for $D_{10,2}$. The accuracy on the backdoor test set is around 0.1 for all data sets.

In Experiment Fed-1.2 we calculate the MNIST accuracy for a 'retraining from scratch' approach, where we remove the target client and train the model with the remaining clients for one and 20 rounds. For IID data we already achieve a relatively high accuracy even without the contribution of the target client (Appendix A.2, Table 10). For mild and strong non-IID data the MNIST accuracy is relatively low after one round of training with a lowest value of 0.104 for $D_{9/10,2}$. After 20 rounds of training, the accuracy for the mild non-IID splits reaches a comparably high value, as in Experiment 1.1.1, where all clients participated in the training process. For strong non-IID data, the accuracy stays lower, with a lowest value of 0.597 for $D_{2/3,1}$.

3.2 Results for Evaluation of Unlearning Experiments

In Experiment Fed-2.1 we evaluate the performance of the original unlearning algorithm for three clients with three different splits, where the target client has a fraction of backdoored data. For all three splits we achieve a high accuracy, both on the MNIST and backdoor test set (Table 2). After unlearning and retraining for one round, the accuracy of the model trained with the IID client split IID_3 remains high at 0.943, while the accuracies of the other two splits drop significantly (0.856 and 0.344). While the backdoor accuracies after unlearning are for all splits lower than before unlearning, they remain higher, than the values achieved with a model, only trained with clean data (cf. Experiment Fed-1.1). The highest remaining backdoor accuracy results for the IID client split (0.502 ± 0.288). Furthermore the backdoor accuracies after unlearning are highly variable, resulting in a standard deviation of more than 0.2 for all client splits.

Training Set Up	**Client Split**	**Train Data**	**MNIST Accuracy**	**MNIST Accuracy$_{UL}$***		**Backdoor Accuracy**	**Backdoor Accuracy$_{UL}$***
				$\mathbf{T_{re} = 1}$	$\mathbf{T_{re} = 20}$		
federated	IID	IID_3	0.99 ± 0.002	0.943 ± 0.027	0.992 ± 0.153	0.997 ± 0.012	0.502 ± 0.288
federated	mildly non-IID	$nIID_{3,2}$	0.987 ± 0.006	0.856 ± 0.017	0.969 ± 0.261	0.988 ± 0.004	0.363 ± 0.223
federated	strongly non-IID	$nIID_{3,1}$	0.968 ± 0.002	0.344 ± 0.056	0.573 ± 0.128	0.948 ± 0.037	0.191 ± 0.215

***** Accuracy$_{UL}$ refers to the accuracy that was calculated after the unlearning process

Table 2: Results for Experiment Fed-2.1 to evaluate the unlearning performance of the original unlearning algorithm, by Halimi et. al, using the target clients local model as unlearning model.

In Experiment Fed-2.2 we we evaluate the performance of our modification to the unlearning algorithm for all possible client splits, where the target client has a fraction of backdoored data. As in Experiment 2.1 both the MNIST and backdoor accuracies before unlearning reach a value of more than 0.9 for all client splits (Table 3).
For the IID client splits the MNIST Accuracy after unlearning and just one round of retraining remains above 0.95 and is therefore higher than in a 'retraining from scratch' approach (Experiment 1.1.2). The backdoor accuracy drops to values comparable to those achieved when training with only clean data (cf. Experiment Fed-2.1).
For mildly non-IID client splits the MNIST accuracy after unlearning remains high for five and ten clients and drops to 0.733 for three clients, still yielding a higher performance than in a 're-training from scratch' approach. Similar to the IID split, the backdoor accuracy after unlearning decreases to values, comparable to those from Experiment Fed-2.1.
For strongly non-IID client splits the MNIST accuracy after unlearning drops significantly but is still higher, than in Experiment 1.1.2. While the backdoor accuracies after unlearning decrease, compared to the values before unlearning, they remain however higher (0.22, 0.212 and 0.189) than the accuracies of a model trained with only clean data.

Training Set Up	**Client Split**	**Train Data**	**MNIST Accuracy**	**MNIST Accuracy$_{UL}$***		**Backdoor Accuracy**	**Backdoor Accuracy$_{UL}$***
				$\mathbf{T_{re} = 1}$	$\mathbf{T_{re} = 20}$		
federated	IID	IID_3	0.991	0.962	0.992	0.996	0.091
		IID_5	0.99	0.961	0.991	0.948	0.113
		IID_{10}	0.959	0.991	0.991	0.991	0.107
federated	mildly non-IID	$nIID_{3,2}$	0.985	0.733	0.981	0.916	0.098
		$nIID_{5,3}$	0.992	0.991	0.991	0.915	0.1
		$nIID_{10,5}$	0.989	0.962	0.99	0.92	0.119
federated	strongly non-IID	$nIID_{3,1}$	0.887	0.594	0.602	0.97	0.222
		$nIID_{5,1}$	0.977	0.636	0.795	0.921	0.212
		$nIID_{10,2}$	0.963	0.634	0.944	0.93	0.189

* Accuracy$_{UL}$ refers to the accuracy that was calculated after the unlearning process

Table 3: Results for Experiment Fed-2.2 to evaluate the unlearning performance of our unlearning algorithm using the global model as unlearning model.

Training Set Up	**Client Split**	**Train Data**	**MNIST Accuracy$_{FS}$****		**MNIST Accuracy$_{UL}$***		**Backdoor Accuracy$_{Cl}$†**	**Backdoor Accuracy$_{UL}$***
			$\mathbf{T_{tr} = 1}$	$\mathbf{T_{tr} = 20}$	$\mathbf{T_{re} = 1}$	$\mathbf{T_{re} = 20}$		
federated	IID	IID_3	0.923	0.989	0.962	0.992	0.102	0.091
		IID_5	0.938	0.991	0.961	0.991	0.101	0.113
		IID_{10}	0.887	0.988	0.991	0.991	0.116	0.107
federated	mildly non-IID	$nIID_{3,2}$	0.319	0.977	0.733	0.981	0.1	0.098
		$nIID_{5,3}$	0.422	0.988	0.991	0.991	0.128	0.1
		$nIID_{10,5}$	0.481	0.982	0.962	0.99	0.084	0.119
federated	strongly non-IID	$nIID_{3,1}$	0.377	0.597	0.594	0.602	0.105	0.222
		$nIID_{5,1}$	0.128	0.792	0.636	0.795	0.102	0.212
		$nIID_{10,2}$	0.104	0.819	0.634	0.944	0.062	0.189

* Accuracy$_{UL}$ refers to the accuracy that was calculated after the unlearning process
** Accuracy$_{FS}$ refers to the accuracy that was calculated in a 'retraining from scratch approach'
† Accuracy$_{Cl}$ refers to the accuracy that was calculated when the model was only trained with clean data

Table 4: Comparison of results from our unlearning algorithm (Experiment Fed-2.2) with a 'retraining from scratch' approach (Experiment Fed-1.2) and training of a model with only clean data (Experiment Fed-1.1)

4 Discussion

The verification of the unlearning process was achieved by the backdoor data. The cut-off for unlearning was established to be 0.1 [17], as this is the accuracy achieved by a model trained on clean data.
As shown in Table 2, the accuracy for the backdoor data of the unlearned model based on the client model is not only much higher, also the standard deviation is very high resulting in unreliable unlearning. Aditionally it can be shown that whilst the accuracy on the backdoor data declines, the more heterogenous the data is, the lower the MNIST accuracy is even after 20 rounds of training. As the model does not relearn the clean data and does not unlearn the data sufficiently, it is not applicable as a secure unlearning process.
Our hypothesis as to why the model did not unlearn the data sufficiently is, that the model did not unlearn "far enough" from the initially trained global model and thus in the relearning of the data, the unlearned model reconverges to the same or a close minimum. This results in an unlearned global model, which is too similar to the global model with the targets client contribu-

tion. Hence, to achieve a model further removed from the global model, we tried unlearning the global model itself, instead of the target clients model, while still using the target clients data. When basing the unlearning model on the global model, the backdoor accuracy of the unlearned model is sufficiently low for the IID and the mildly non-IID data. This unlearned model also shows a high accuracy on the MNIST dataset. Especially after 20 rounds, the accuracy is as high as before the unlearning. Additionally, when comparing the MNIST accuracy after only one round of learning with the baseline for the 'retrain from scratch' approach, one can observe that the accuracy for the unlearned model is consistently higher by 0.04 up to 0.4 with an even higher difference when comparing the mildly non-IID data. With the accuracy of the first retraining round of the global unlearned model being higher than the accuracy of the first retraining round of the local unlearned model, one can assume a faster convergence and an even faster algorithm than presented in the paper. Thus it can be assumed that this model is not only unlearning IID and mildly non-IID data reliably, but also outputs a model much faster than it would be to retrain the model on the data. Nevertheless our experiments show, that the algorithm unfortunately does not produce reliable results for strongly non-IID data, therefor different optimizations of the algorithm need to be considered.

5 Outlook

Further interesting features might be investigated further to enhance the algorithm.
Preliminary tests were run, with an increased τ the early stopping criterion, to hopefully decrease the time of the unlearning process and the subsequent relearning process. As seen in Appendix A.2, Table 11, one can see that the accuracy on the backdoor data was sufficiently low, while the accuracy on the MNIST data is very high. This however was only tested on IID data and has yet to be tested for non-IID data.
Furthermore, the applicability of this algorithm for unlearning smaller datasets might be interesting. As we now use the entirety of the target clients dataset to unlearn the model from the backdoor data, it might be interesting to only unlearn the backdoor data and use the remaining clean data to also retrain the model.

References

[1] P. Rajpurkar, E. Chen, O. Banerjee, and E. J. Topol, "Ai in health and medicine," *Nature Medicine*, vol. 28, no. 1, pp. 31–38, 2022.

[2] M. J. Sheller, B. Edwards, G. A. Reina, J. Martin, S. Pati, A. Kotrotsou, M. Milchenko, W. Xu, D. Marcus, R. R. Colen, *et al.*, "Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data," *Scientific reports*, vol. 10, no. 1, pp. 1–12, 2020.

[3] European Commission, "General data protection regulation (GDPR)," 2016. `https://gdpr.eu/tag/gdpr/`, Last accessed on 11.01.2023.

[4] C. Dwork, A. Roth, *et al.*, "The algorithmic foundations of differential privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014.

[5] A. Cheu, A. Smith, J. Ullman, D. Zeber, and M. Zhilyaev, "Distributed differential privacy via shuffling," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 375–403, Springer, 2019.

[6] J. Dong, A. Roth, and W. J. Su, "Gaussian differential privacy," *arXiv preprint arXiv:1905.02383*, 2019.

[7] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: Theory and implementation," *ACM Computing Surveys (Csur)*, vol. 51, no. 4, pp. 1–35, 2018.

[8] C. Zhao, S. Zhao, M. Zhao, Z. Chen, C.-Z. Gao, H. Li, and Y.-a. Tan, "Secure multi-party computation: theory, practice and applications," *Information Sciences*, vol. 476, pp. 357–372, 2019.

[9] K. A. Jagadeesh, D. J. Wu, J. A. Birgmeier, D. Boneh, and G. Bejerano, "Deriving genomic diagnoses without revealing patient genomes," *Science*, vol. 357, no. 6352, pp. 692–695, 2017.

[10] B. McMahan, E. Moor, D. Range, S. Hampson, and B. Agueeray Arcas, "Communication-efficient learning of deep networks from decentralized data," *WCP*, 2017.

[11] J. Konecny, B. McMahan, D. Ramage, and P. Richtarik, "Federated optimization: Distributed machine learning for on-device intelligence," *WCP*, 2017.

[12] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, J. Fernandez-Marques, Y. Gao, L. Sani, K. H. Li, T. Parcollet, P. P. B. de Gusmão, and N. D. Lane, "Flower: A friendly federated learning research framework," *arXiv*, 2020.

[13] G. A. Reina, A. Gruzdev, P. Foley, O. Perepelkina, M. Sharma, I. Davidyuk, I. Trushkin, M. Radionov, A. Mokrov, D. Agapov, J. Martin, B. Edwards, M. J. Sheller, S. Pati, P. N. Moorthy, H. S. Wang, P. Shah, and S. Bakas, "Openfl: An open-source framework for federated learning," *CoRR*, 2021.

[14] J. Matschinske, J. Späth, R. Nasirigerdeh, R. Torkzadehmahani, A. Hartebrodt, B. Orbán, S. Fejér, O. Zolotareva, M. Bakhtiari, B. Bihari, M. Bloice, N. C. Donner, W. Fdhila, T. Frisch, A.-C. Hauschild, D. Heider, A. Holzinger, W. Hötzendorfer, J. Hospes, T. Kacprowski, M. Kastelitz, M. List, R. Mayer, M. Moga, H. Müller, A. Pustozerova, R. Röttger, A. Saranti, H. H. Schmidt, C. Tschohl, N. K. Wenke, and J. Baumbach, "The featurecloud ai store for federated learning in biomedicine and beyond," 2021.

[15] G. Liu, X. Ma, Y. Yang, C. Wang, and J. Liu, "Federaser: Enabling efficient client-level data removal from federated learning models," *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*, 2021.

[16] C. Wu, S. Zhu, and P. Mitra, "Federated unlearning with knowledge distillation," *CoRR*, vol. abs/2201.09441, 2022.

[17] A. Halimi, S. Kadhe, A. Rawat, and N. Baracaldo, "Federated unlearning: How to efficiently erase a client in fl?," *arXiv preprint arXiv:2207.05521*, 2022.

[18] C. Guo, T. Goldstein, A. Hannun, and L. van der Maaten, "Certified data removal from machine learning models," *arXiv*, 2019.

[19] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," *arXiv*, 2012.

[20] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[21] D. P. Kingma, J. A. Ba, and J. Adam, "A method for stochastic optimization. arxiv 2014," *arXiv preprint arXiv:1412.6980*, vol. 106, 2020.

A Appendix

A.1 Details on hyperparameters and datasets used for the experiments

Exp. Nr.	**Training Set Up**	**Train Data**	**Batch Size**	**Epochs**	$\mathbf{T_{tr}}$
Cent-1.1	Centralized	{D}	128	10	-

Fed-1.1	Federated	{IID$_3$,IIDD$_5$, IID$_{10}$}
		{nIID$_{3,2}$, nIID$_{5,3}$, nIID$_{10,5}$}
		{nIID$_{3,1}$, nIID$_{5,1}$}
			128	-	20

		{nIID$_{10,2}$}	128	-	30

Table 5: Experimental details for baseline comparison of centralized and federated learning.

Exp. Nr.	**Training Set Up**	**Train Data***	**Batch Size**	$\mathbf{T_{tr}}$

Fed-1.2	Federated	{IID$_{2/3}$,IID$_{4/5}$, IID$_{9/10}$}
		{nIID$_{2/3,2}$, nIID$_{4/5,3}$, nIID$_{9/10,5}$}
		{nIID$_{2/3,1}$, nIID$_{4/5,1}$, nIID$_{9/10,2}$}
			128	1

				20

* The notation of the data set in these experiments is (n)IID$_{x/N,k}$, where N denotes the initial number of clients, the data was split among and x the number of clients that participated in the 'retraining from scratch' approach.

Table 6: Experimental details for baseline comparison of federated learning from scratch and federated unlearning

Exp. Nr.	**Training Set Up**	**Train Data**	$\mathbf{p}_{BD}$	**Unlearning Model**	**Batch Size**	$\mathbf{T_{tr}}$	$\mathbf{T_{re}}$
Fed-2.1	Federated	{IID$_3$,nIID$_{3,2}$, nIID$_{3,1}$}	0.3	local client model	128	40	1
							20

Table 7: Experimental details for Federated Unlearning with local client model

Exp. Nr.	**Training Set Up**	**Train Data**	$\mathbf{p}_{BD}$	**Unlearning Model**	**Batch Size**	$\mathbf{T_{tr}}$	$\mathbf{T_{re}}$
Fed-2.2	Federated	{IID$_3$, nIID$_{3,2}$, nIID$_{3,1}$}	0.3	global model	128	40	1
							20
		{IID$_5$,nIID$_{5,3}$, nIID$_{5,1}$}	0.66				1
							20
		{IID$_{10}$,nIID$_{10,5}$, nIID$_{10,2}$}	0.8				1
							20

Table 8: Experimental details for Federated Unlearning with global model

A.2 Datailled Results for Experiments

Training Set Up	**Client Split**	**Train Data**	**MNIST Accuracy**	**Backdoor Accuracy**
centralized	-	D	0.991	0.101
federated		IID_3	0.99	0.102
federated	IID	IID_5	0.991	0.101
federated		IID_{10}	0.99	0.116
federated		$\text{nIID}_{3,2}$	0.989	0.1
federated	mildly non-IID	$\text{nIID}_{5,3}$	0.989	0.128
federated		$\text{nIID}_{10,5}$	0.984	0.084
federated		$\text{nIID}_{3,1}$	0.971	0.105
federated	strongly non-IID	$\text{nIID}_{5,1}$	0.962	0.102
federated		$\text{nIID}_{10,2}$	0.955	0.062

Table 9: Results for baseline experiments (Exoeriment Cent-1.1, Fed-1.1), comparing MNIST accuracy and backdoor accuracy for centralized and federated learning

Training Set Up	**Client Split**	**Train Data**	**MNIST Accuracy**	
			$\mathbf{T_{tr} = 1}$	$\mathbf{T_{tr} = 20}$
federated		$\text{IID}_{2/3,3}$	0.923	0.989
federated	IID	$\text{IID}_{4/5,5}$	0.938	0.991
federated		$\text{IID}_{9/10,10}$	0.887	0.988
federated		$\text{nIID}_{2/3,2}$	0.319	0.977
federated	mildy non-IID	$\text{nIID}_{4/5,3}$	0.422	0.988
federated		$\text{nIID}_{9/10,5}$	0.481	0.982
federated		$\text{nIID}_{2/3,1}$	0.377	0.597
federated	strongly non-IID	$\text{nIID}_{4/5,1}$	0.128	0.792
federated		$\text{nIID}_{9/10,2}$	0.104	0.819

Table 10: Results for Experiment Fed-1.2, determining MNIST accuracy for a 'retrain from scratch' approach after one and 20 rounds

Training Set Up	**Client Split**	**Train Data**	**MNIST Accuracy$_{UL}$***		**Backdoor Accuracy$_{UL}$***	
			$\tau = 0.12$	$\tau = 0.5$	$\tau = 0.12$	$\tau = 0.5$
federated	IID	IID_3	0.91 ± 0.04	0.986 ± 0.003	0.04 ± 0.043	0.04 ± 0.046

* Accuracy$_{UL}$ refers to the accuracy that was calculated after the unlearning process

Table 11: Results for the unlearning algorithm using the global model as unlearning model with different early stopping criteria. The model was trained for $T_{tr} = 20$ for a total of 30 runs

A.3 Data Distributions

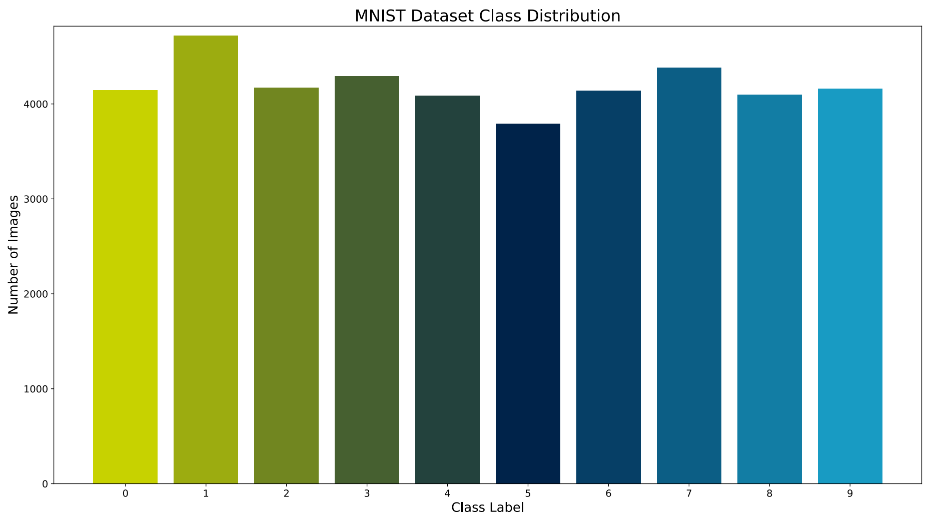

Figure 1: Class Distribution of the entire MNIST train data set

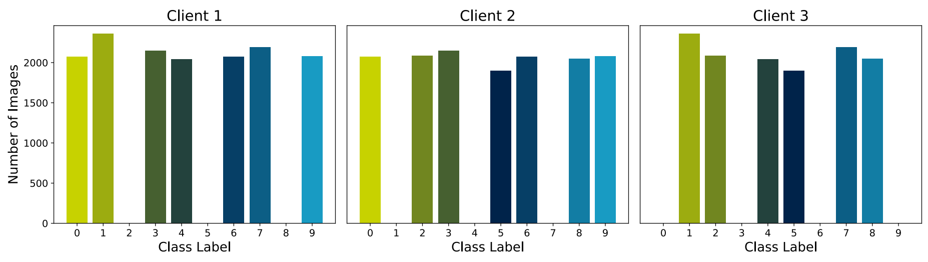

Figure 2: Class Distribution for mildly non-IID client split among three clients $\mathrm{nIID}_{3,2}$

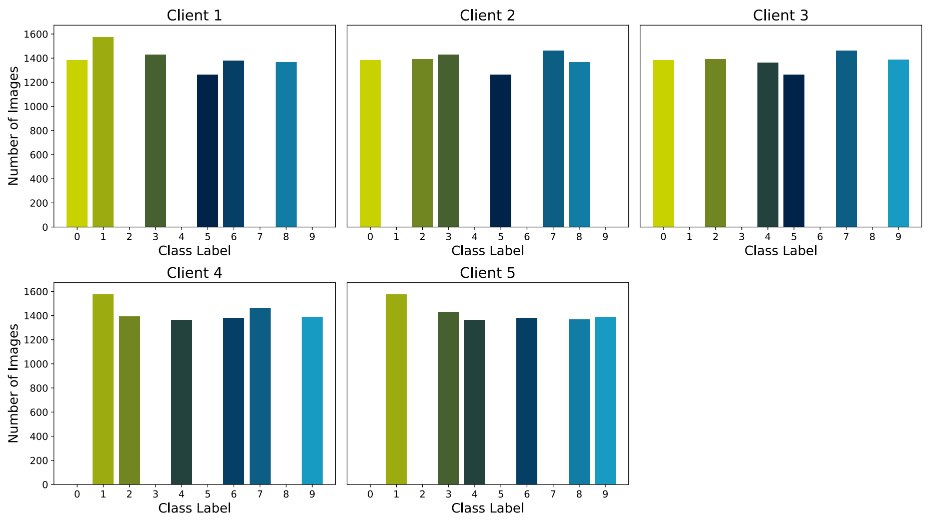

Figure 3: Class Distribution for mildly non-IID client split among five clients $\mathrm{nIID}_{5,3}$

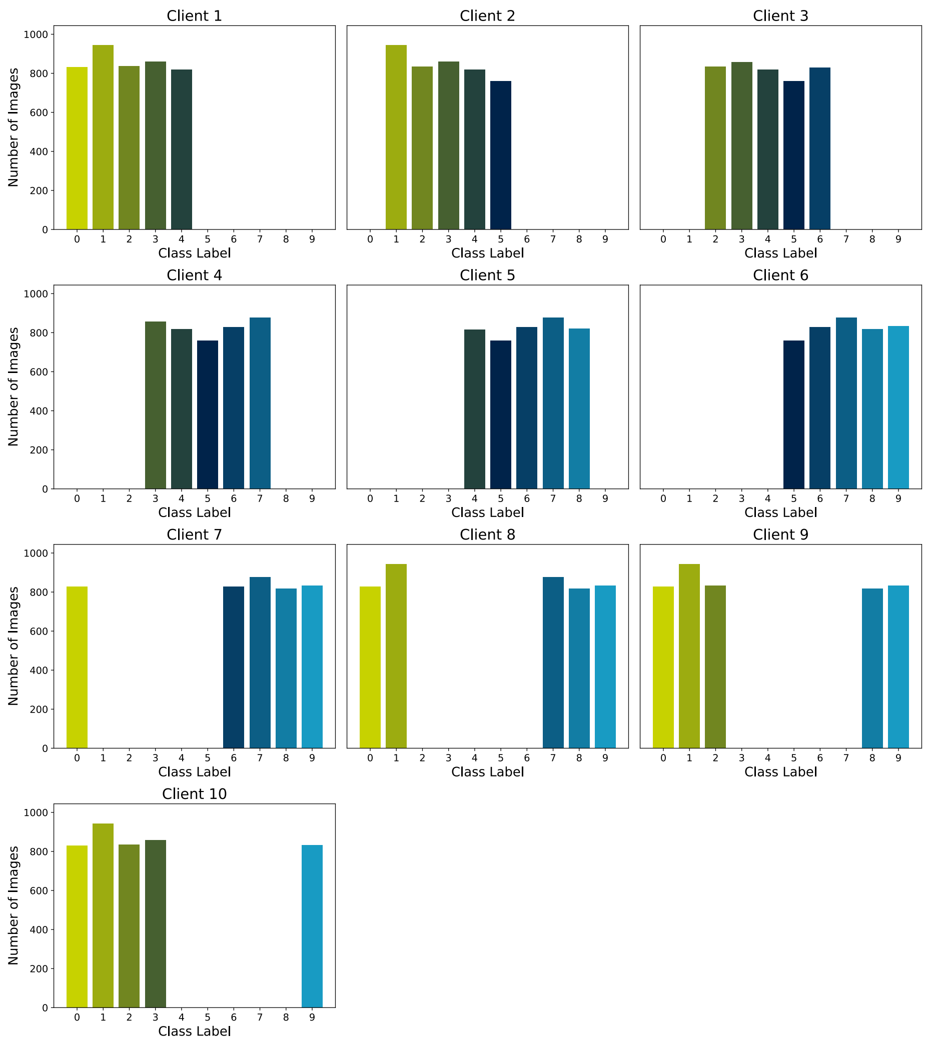

Figure 4: Class Distribution for mildly non-IID client split among ten clients nIID$_{10,5}$

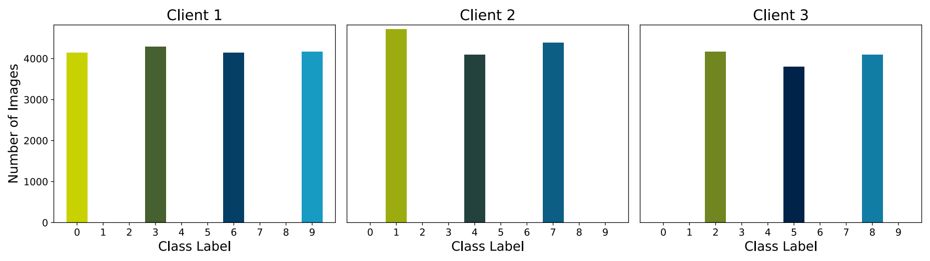

Figure 5: Class Distribution for strongly non-IID client split among three clients nIID$_{3,2}$

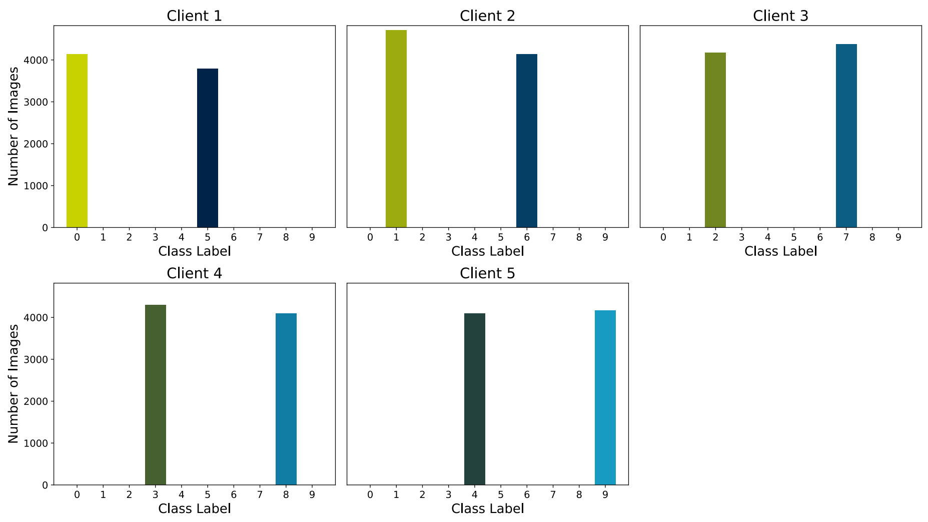

Figure 6: Class Distribution for strongly non-IID client split among five clients $\text{nIID}_{5,3}$

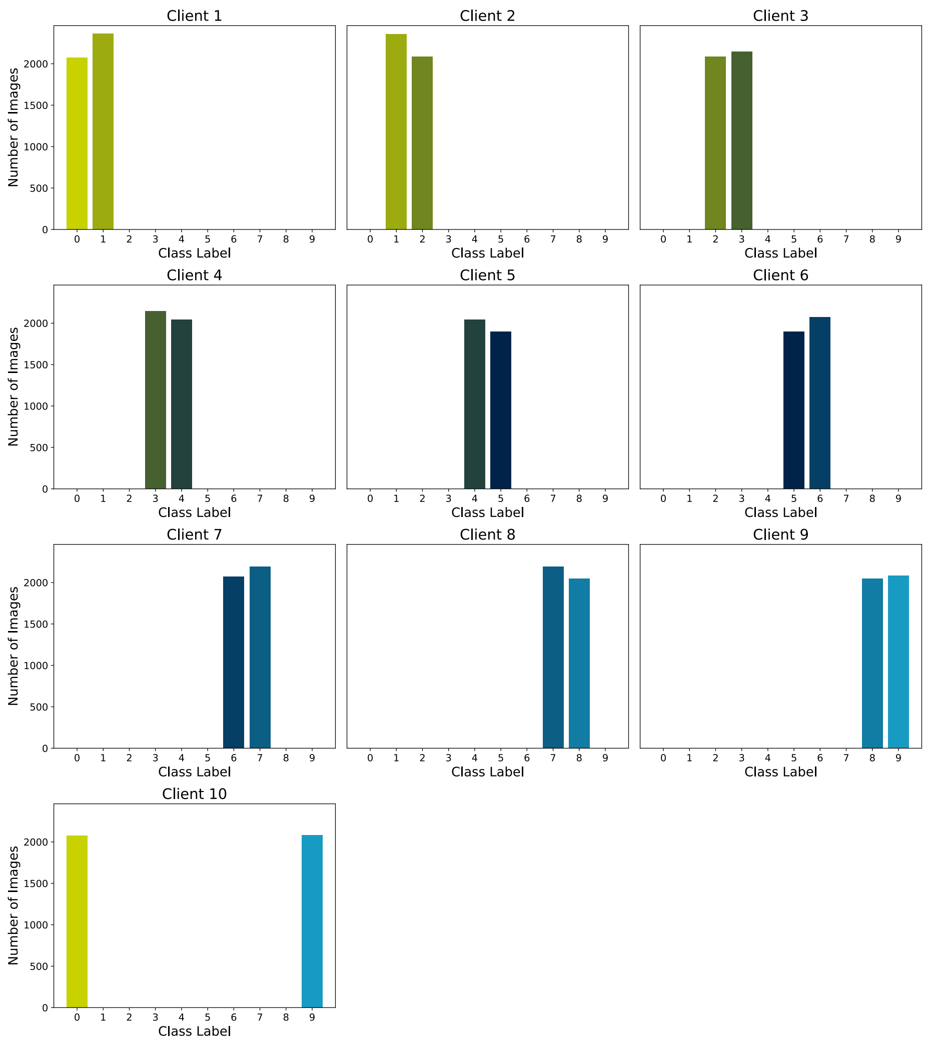

Figure 7: Class Distribution for strongly non-IID client split among ten clients $\text{nIID}_{10,5}$

B Implementation

While traditional Machine Learning (ML) algorithms are executed locally, Federated Learning (FL) requires an infrastructure to enable exchange of parameters between collaborating parties. In addition to computation on each of the participants' devices, a coordinator needs to be involved to aggregate different models and facilitate communication between the clients. The FeatureCloud platform [14] was developed to overcome these hurdles of implementing FL algorithms. FeatureCloud provides two classes of frameworks to work with: in the FeatureCloud AI Store, users without programming experience have access to a graphical user interface for a variety of ML models, including cross-validation, linear regression and random forest. Additionally, for experienced programmers there are backend frameworks that come with methods to implement FL algorithms more easily.

Using this backend-only approach, we developed an application for unlearning using the algorithm described above. While this application in itself does not perform in a federated fashion, as it loads a previously trained model ('global model') and unlearns it with the target client's data ('unclient model') using a projected gradient ascent, it is suited and intended to be integrated in a federated workflow.

The application starts in an initial state, where a config file is read to control parameters, such

as the number of clients and the number of classes, as well as hyper parameters, such as learning rate and weight decay of the employed optimizer. The global model that will be unlearned and pairs of images and labels from the target client are also loaded in this state. After transitioning to the computation state, data of the target client is split into a training set and validation set to build the data loaders. Over a set number of epochs, the global model is then unlearned as explained in 2. In the writing state, the untrained global model is exported and incorporated in the workflow downstream, where it is relearned with data of the remaining clients.
Link to the GitHub: https://github.com/AntoniaGocke/dumplings

C Model Architecture

Layer number	Layer type	Dimension	Stride	Padding	Dropout	Activation
1	Convolution	(8x5x5)	2	2	0.2	SiLU
2	Convolution	(8x5x5)	2	2	0.2	SiLU
3	Convolution	(16x5x5)	2	2	0.2	SiLU
4	Convolution	(32x5x5)	2	2	-	SiLU
5	Fully Connected	(256)	-	-	-	SiLU
6	Fully Connected	(10)	-	-	-	Softmax

Table 12: CNN model architecture