Federated Unlearning with Gradient Descent and Conflict Mitigation

Anonymous submission

Abstract

Federated Learning (FL) has received much attention in recent years. However, although clients are not required to share their data in FL, the global model itself can implicitly remember clients' local data. Therefore, it's necessary to effectively remove the target client's data from the FL global model to ease the risk of privacy leakage and implement "the right to be forgotten". Federated Unlearning (FU) has been considered a promising way to remove data without full retraining. But the model utility easily suffers significant reduction during unlearning due to the gradient conflicts. Furthermore, when conducting the post-training to recover the model utility, the model is prone to move back and revert what has already been unlearned. To address these issues, we propose Federated Unlearning with Orthogonal Steepest Descent (FedOSD). We first design an unlearning Cross-Entropy loss to overcome the convergence issue of the gradient ascent. A steepest descent direction for unlearning is then calculated in the condition of being non-conflicting with other clients' gradients and closest to the target client's gradient. This benefits to efficiently unlearn and mitigate the model utility reduction. After unlearning, we recover the model utility by maintaining the achievement of unlearning. Finally, extensive experiments in several FL scenarios verify that FedOSD outperforms the SOTA FU algorithms in terms of unlearning and model utility.

1 Introduction

Federated Learning (FL) has increasingly gained popularity as a machine learning paradigm in recent years (McMahan et al. 2017). It allows clients to cooperatively train a global model without sharing their local data, which helps address data island and privacy issues (Yu et al. 2022). But previous studies demonstrate that clients' local training data is inherently embedded in the parameter distribution of the models trained on it (De and Pedersen 2021; Zhao et al. 2023). Therefore, in light of privacy, security, and legislation issues, it's necessary to remove clients' training data from the trained model (Zhang et al. 2023), especially when clients opt to withdraw from FL. This is known as the right to be forgotten (RTBF) (Liu et al. 2021), which is enacted by privacy regulations such as the General Data Protection Regulation (GDPR) (Voigt and Von Bussche 2017) and the California Consumer Privacy Act (CCPA) (Harding et al. 2019).

A naive way to achieve this goal is to retrain the FL model. But it brings large computation and communication costs (Liu et al. 2023). In contrast, unlearning is a more efficient way, which has been well studied in centralized machine learning (Bourtoule et al. 2021). Inspired by it, Federated Unlearning (FU) has emerged, aiming to remove data from a trained FL model while trying to maintain model utility.

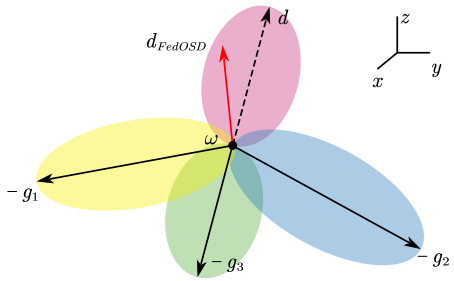

Figure 1: A demo of three clients. g_1, g_2, g_3 represent the gradient of clients. d denotes the update direction for unlearning client 3, which is conflicting with g_1 and g_2, i.e., $g_1 \cdot d < 0$ and $g_2 \cdot d < 0$. d_{FedOSD} represents the direction obtained by FedOSD, which doesn't conflict with g_1 and g_2.

In this context, numerous FU techniques have been proposed. Federaser (Liu et al. 2021) leverages the norms of historical local updates in the previous FL training to accelerate retraining. FedKdu (Wu, Zhu, and Mitra 2022) and FedRecovery (Zhang et al. 2023) utilize the historical gradients to calibrate the model to erase the training data of the target client (i.e., the client that requests for unlearning). However, these methods require clients to continuously record historical information during FL training (Yang and Zhao 2023). Moreover, (Zhao et al. 2023) propose MoDe to unlearn the target client's data by momentum degradation, but it requires simultaneously retraining a model for updating the unlearning model, which brings additional communication costs.

Among prior studies, Gradient Ascent (GA) is considered a viable and efficient method for FU (Liu et al. 2023), which formulates unlearning as the inverse process of learning and takes the inverse of the loss function to reduce the model performance on the target client. It can effectively achieve the unlearning goal in few communication rounds while not bringing extra storage costs (Halimi et al. 2022). However, we observe that there exist the following three primary challenges when performing GA in FU.

Challenge 1: Gradient explosion. Gradient explosion is a significant challenge for GA-based federated unlearning, necessitating a substantial reliance on experimental hyperparameter tuning. This is because the loss function generally has no upper bound (see Fig. 3(a)). Consequently, executing GA to unlearn results in gradient explosion and cannot converge. We delve further into this in Section 3.1. To this end, (Halimi et al. 2022) project model parameters to an L_2-norm ball of radius δ. But it requires experimentally tuning δ.

Challenge 2: Model utility degradation. Directly applying GA to unlearn would inevitably destroy the model utility (Yang and Zhao 2023), even leading to catastrophic forgetting (Liu et al. 2023). Specifically, the model performance on remaining clients (i.e., those that do not require unlearning) would decrease heavily. One direct cause is the gradient conflict (Pan et al. 2023), where the model update direction for unlearning a client conflicts with those of the remaining clients, directly leading to a reduction in model utility. Fig. 1 illustrates an example in which client 3 requests unlearning, but the model update direction conflicts with the gradients of client 1 and client 2. Consequently, the updated model would exhibit diminished performance on client 1 and 2.

Challenge 3: Model reverting issue in post-training. After unlearning, post-training is often conducted, where the target client leaves and the remaining clients continually train the FL global model cooperatively to recover the model utility that was reduced in the previous unlearning (Halimi et al. 2022; Wu, Zhu, and Mitra 2022). However, we observe that during this stage, the model tends to revert to its original state, resulting in the recovery of previously forgotten information and thus losing the achievement of unlearning. This issue is further explored in Section 3.3.

To handle the aforementioned challenges, we propose the **Fed**erated Unlearning with **O**rthogonal **S**teepest **D**escent algorithm (FedOSD). Specifically, to handle the gradient explosion inherent in GA, we modify the Cross-Entropy loss to an unlearning version and employ the gradient descent, rather than GA, to achieve the unlearning goal. Subsequently, an orthogonal steepest descent direction that avoids conflicts with retained clients' gradients is calculated to better unlearn the target client while mitigating the model utility reduction. In post-training, we introduce a gradient projection strategy to prevent the model from reverting to its original state, thereby enabling the recovery of model utility without compromising the unlearning achievement.

Our contributions are summarized as follows:

1. We introduce an Unlearning Cross-Entropy loss that can overcome the convergence issue of Gradient Ascent.
2. We propose FedOSD that establishes an orthogonal steepest descent direction to accelerate the unlearning process while mitigating the model utility reduction.
3. We design a gradient projection strategy in the post-training stage to prevent the model from reverting to its original state for better recovering the model utility.
4. We implement extensive experiments on multiple FL scenarios, validating that FedOSD outperforms the SOTA FL unlearning approaches in both unlearning performance and the model utility.

2 Background & Related Work

2.1 Federated Learning (FL)

The traditional FL trains a global model ω cooperatively by m clients, which aims to minimize the weighted average of their local objectives (Li et al. 2020): $\min_\omega \sum_{i=1}^{m} p_i L_i(\omega)$, where $p_i \geq 0$, $\sum_{i=1}^{m} p_i = 1$. L_i is the local objective of client i, which is usually defined by the empirical risks over the local training data with N_i samples: $L_i(\omega^t) = \sum_{j=1}^{N_i} \frac{1}{N_i} L_{i_j}(\omega^t)$. L_{i_j} is the loss on sample j, which is obtained by a specific loss function such as Cross-Entropy (CE) loss:

$$L_{CE} = -\sum_{c=1}^{C} y_{o,c} \cdot log(p_{o,c}), \tag{1}$$

where C denotes the number of classes. $y_{o,c}$ is the binary indicator (0 or 1) if class label c is the correct classification for observation o, i.e., the element of the one-hot encoding of sample j's label. $p_{o,c}$ represents the predicted probability observation o that is of class c, which is the o^{th} element of the softmax result of the model output.

2.2 Federated Unlearning

Federated Unlearning (FU) aims to erase the target training data learned by the FL global model, while mitigating the negative impact on the model performance (e.g., accuracy or local objective). Recognized as a promising way to protect 'the Right to be Forgotten' of clients, FU can also counteract the impact of data poisoning attacks to enhance the security (Yang and Zhao 2023; Liu et al. 2023).

FU has garnered increasing interest in recent years. (1) Some previous studies have leveraged the historical information of FL training to ease the target client's training data, such as FedEraser (Liu et al. 2021), FedKdu (Wu, Zhu, and Mitra 2022), FedRecovery (Zhang et al. 2023), etc. (2) Besides, (Zhao et al. 2023) adopt momentum degradation to FU. (3) (Su and Li 2023) use clustering and (4) (Ye et al. 2024) employ distillation to unlearn. (5) A significant approach related to our work is Gradient Ascent, which utilizes the target client's gradients for unlearning (Halimi et al. 2022; Wu et al. 2022).

Based on the types of client data that need to be forgotten, Federated Unlearning can be categorized into sample unlearning and client unlearning (Liu et al. 2023). We focus on client unlearning in this paper for two reasons. First, we can make a fair comparison with previous record-based FU methods such as FedEraser and FedRecovery. Since they rely on pre-recording information like model gradients on the target data that needed to be unlearned, they are not suitable for sample unlearning. Since in the sample unlearning, clients only request to unlearn partial training data. However, no one knows which data will be requested to unlearn during FL training, and thus preparing these records in advance for later unlearning is not feasible in practice.

Furthermore, for other FU algorithms that do not necessitate using historical training records, we can technically treat unlearning samples as belonging to a virtual client. Hence, the sample unlearning can be transferred to the client unlearning. For example, when a client requests to unlearn partial data D_u, we can form a new virtual client u that owns D_u and unlearn it.

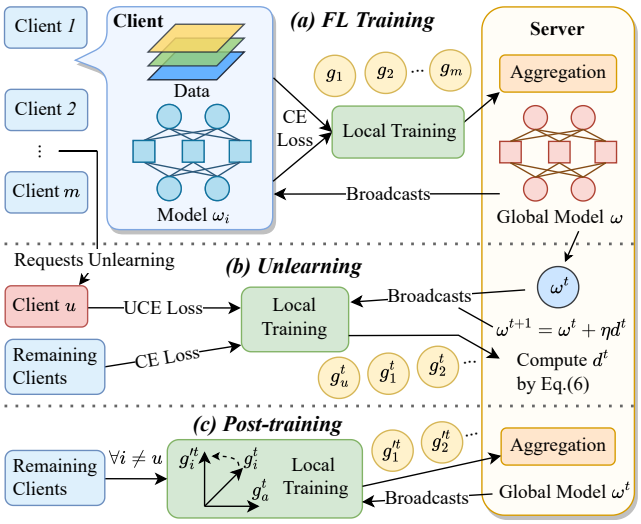

Figure 2: The FedOSD framework comprises two main stages: (b) the unlearning stage and (c) the post-training stage. Subfigure (a) depicts the previous FL training procedure before the client requests for unlearning, where the obtained model is denoted as ω^0 and serves as the original model for unlearning.

The formulation of unlearning the target client u from the trained global model can be defined by:

$$\max_{\omega} \; L_u(\omega), \tag{2}$$

where $L_u(\omega)$ represents the local objective of client u in FL.

Federated Unlearning with Gradient Ascent. Gradient Ascent (GA) (Wu et al. 2022; Liu et al. 2023) is a proactive and efficient approach for solving Problem (2). At each communication round t, it strives to maximize the empirical loss of the target client u by updating the model according to $\omega^{t+1} = \omega^t + \eta^t \nabla L_u(\omega^t)$ with the step size (learning rate) η^t (Halimi et al. 2022). However, (Wu et al. 2022) suggest that this approach would fail because of destroying the global model performance for the remaining clients. To this end, they propose EWCSGA, which incorporates a regularization term to the cross entropy loss to mitigate the negative impact on the model utility. Besides, SFU (Li et al. 2023) projects the gradient of the target client to a vector that is orthogonal to the subspace formed by the remaining clients' representation matrices. This approach, however, entails substantial communication costs and raises privacy concerns as it requires each client to upload the output of each model layer after performing a forward propagation on each selected data sample to the server.

Our method draws from the idea of GA to achieve the goal of unlearning. Differently, we modify the CE loss function to an unlearning version to overcome the gradient explosion issue, and compute the steepest descent direction that not only aligns closely with the target client's gradient but also avoids conflicts with the retained clients' gradients. This approach enables more effective unlearning while mitigating the model utility degradation.

Algorithm 1: **FedOSD**

Input: Pretrained model ω^0, learning rate η, FL client set S, communication round T, max unlearning round T_u.

$u \in S \leftarrow$ The client requests for unlearning.
for $t = 0, 1, \cdots, T_u - 1$ **do**
  Server broadcasts ω^t to all client $i \in S$.
  $\omega_i^t \leftarrow$ Each client i performs local training, in which client u switches to utilize UCE loss (Eq. (3)).
  Server receives $g_i^t = (\omega^t - \omega_i^t)/\eta$ from each client i.
  $G \leftarrow \text{concat}(g_1^t, \cdots, g_i^t, \cdots), \forall i \in S, i \neq u$.
  Calculate orthogonal steepest direction d^t by Eq. (6).
  $\omega^{t+1} \leftarrow \omega^t + \eta d^t$.
$S \leftarrow S \backslash u$, and start the post-training stage.
for $t = T_u, T_u + 1, \cdots, T$ **do**
  The server broadcasts ω^t to all client $i \in S$.
  Each client i performs local training to obtain g_i^t.
  $g_a^t \leftarrow \nabla_{\omega^t} \frac{1}{2} \|\omega^t - \omega^0\|^2$.
  if $g_i^t \cdot g_a^t > 0$ **then**
    $g_i'^t \leftarrow$ Project g_i^t to the normal plane of g_a^t.
    Rescale $g_i'^t$ by $g_i'^t \leftarrow g_i'^t / \|g_i'^t\| \cdot \|g_i^t\|$.
  else
    $g_i'^t \leftarrow g_i^t$.
  Server receives $g_i'^t$ and aggregates $\bar{g}'^t = \frac{1}{|S|} \sum_i g_i'^t$.
  $\omega^{t+1} \leftarrow \omega^t - \eta \bar{g}'^t$.

Output: Model parameters ω^t.

3 The Proposed Approach

Our proposed FedOSD aims to effectively remove the target client's data from the FL global model while mitigating the model performance reduction across remaining clients. Fig. 2 demonstrates the framework of FedOSD, which includes two stages: unlearning (Fig. 2(b)) and post-training (Fig. 2(c)). ω^0 is the global model previously trained through Federated Learning across m clients (Fig. 2(a)). When client u requests for unlearning, it utilizes the proposed Unlearning Cross-Entropy loss to conduct the local training. After collecting local gradients g_i^t, the server calculates a direction d^t that is closest to client u's gradient while orthogonal to remaining clients' gradients, and then updates the model by $\omega^{t+1} = \omega^t + \eta^t d^t$. In the post-training stage, a gradient projection strategy is performed to prevent the model from reverting to ω^0. Detailed steps of FedOSD can be seen in Algorithm 1. In Appendix.A.2, we prove the convergence of FedOSD in the unlearning and post-training stages.

3.1 Unlearning Cross-Entropy Loss

We first take a brief review of how Gradient Ascent can drive the model to unlearn. As shown in Fig. 3(a), by updating the global model with $\omega^{t+1} = \omega^t + \eta \nabla L_u(\omega^t)$, the local loss increases and $p_{o,c}$ approaches 0, thus degrading the model's prediction accuracy on the target client's data and achieving unlearning. However, the CE Loss (Eq.(1)) has no upper bound. As seen in Fig. 3(a), when $p_{o,c}$ is getting quite close to 0, $\partial L_{CE}/\partial p_{o,c}$ would suffer the explosion and thus the local gradient of the target unlearning client explodes. That's why directly applying GA to unlearn would make the

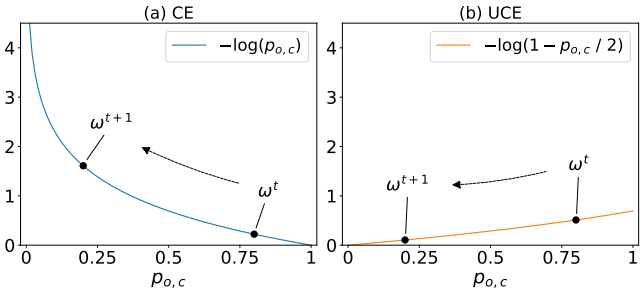

Figure 3: A comparison between (a) Cross-Entropy and (b) the proposed Unlearning Cross-Entropy. When using CE loss and GA to unlearn, it needs to drive $p_{o,c}$ to 0, leading to gradient explosion and non-convergence. When the target client switches to utilize UCE, it adopts the gradient descent to drive $p_{o,c}$ to 0 and wouldn't bring the convergence issue.

model similar to a random model (Halimi et al. 2022). One conventional solution is to project the model back to an L_2-norm ball of radius δ (Halimi et al. 2022). But it brings a hyper-parameter that requires experimentally tuning, and a fixed δ cannot guarantee the convergence.

To address this issue, we modify CE loss to an unlearning version named Unlearning Cross-Entropy (UCE) loss:

$$L_{UCE} = -\sum_{c=1}^{C} y_{o,c} \cdot log(1 - p_{o,c}/2). \tag{3}$$

By minimizing Eq.(3), we can drive the predicted probability $p_{o,c}$ to be closer to 0 (as seen in Fig. 3(b)), thereby diminishing the prediction ability of the model on the target client's data to unlearn it. Note that before unlearning, the model ω^0 often performs well on clients, where $p_{o,c}$ is close to 1 and the model update step for the global model is already quite small. Hence, the constant "2" in Eq.(3) is set to ensure that the gradient norm of the target client does not exceed those of the remaining clients. This can prevent the unlearning process from being unstable or even directly damaging the model utility. We verify this in Appendix.B.2.

Hence, when client u requests for unlearning, it no longer applies GA on the CE loss. Instead, it switches to utilize UCE loss and performs gradient descent to train the model. Since UCE loss has the lower bound 0, it can achieve the goal of unlearning client u's data without bringing issues of gradient explosion and convergence difficulties. Denote $\tilde{L}_u$ as the local objective of the target client u by using UCE loss, then the unlearning formulation (2) is transferred to:

$$\min_{\omega} \tilde{L}_u(\omega). \tag{4}$$

3.2 Orthogonal Steepest Descent Direction

In FedOSD, we solve Problem (4) to unlearn the target client u by iterating $\omega^{t+1} = \omega^t + \eta^t d^t$, where d^t is an orthogonal steepest descent direction at t^{th} round. In this section, we discuss how to obtain such an update direction and analyze how it can accelerate unlearning while mitigating the negative impact on the model utility. We start by introducing the gradient conflict, which is a direct cause of model performance degradation on FL clients (Wang et al. 2021).

Definition 1 (Gradient Conflict): The gradients of client i and j are in conflict with each other iff $g_i \cdot g_j < 0$.

In each communication round t, denote $g_i^t, i \neq u$ as the local gradient of remaining clients, and g_u^t as the gradient of the target client u for unlearning. If we directly adopt $-g_u^t$ as the direction to update the model to unlearn client u, i.e., $\omega^{t+1} = \omega^t - \eta^t g_u^t$, the model performance on the remaining clients would easily suffer reduction because g_u^t would conflict with some g_i^t. The experimental results of Table 2 corroborate the presence and the impact of such gradient conflicts in FU.

Hence, mitigating gradient conflicts can help alleviate decreases in the model utility. One ideal solution would be identifying a common descent direction d^t that satisfies $d^t \cdot g_u^t < 0$ and $d^t \cdot g_i^t < 0$. However, such a strategy could lead to the model becoming prematurely trapped in a local Pareto optimum, which remains far from the optimum of Problem (4). We verify it in the ablation experiments (Section 4.3).

To this end, we mitigate the gradient conflict by computing a model update direction d^t orthogonal to the gradient of the remaining clients, i.e., $d^t \cdot g_i^t = 0, \forall i \neq u$. Although d^t is not a common descent direction, it helps slow down the performance reduction of the model on the remaining clients. However, in FL, the number of remaining clients (i.e., $m-1$) is significantly smaller than D (the dimension of model parameters), implying $\text{rank}(\forall g_i^t, i \neq u) \leq m - 1 << D$. Consequently, there are numerous orthogonal vectors d that satisfy $d \cdot g_i^t = 0$. Therefore, if the obtained direction differs significantly from $-g_u^t$, it would impede the unlearning process and potentially exacerbate the degradation of model utility. We verify it in the ablation study in Section 4.3.

Denote $G \in \mathbb{R}^{(m-1)\times D}$ as a matrix where each row represents a gradient of a remaining client, the key idea is to find a d^t that satisfies $Gd^t = \vec{0}$ while being closest to $-g_u^t$ to accelerate unlearning, i.e., $d^t = \arg\min_{d^t} cos(g_u^t, d^t)$. To maintain the direction's norm, we fix $\|d^t\| = \|g_u^t\|$, then the problem is equivalent to:

$$\begin{aligned} &\min_{d^t \in \mathbb{R}^D} \ \frac{g_u^t \cdot d^t}{\|g_u^t\|^2}, \\ &s.t. \ \ Gd^t = \vec{0}, \\ &\quad \|d^t\| = \|g_u^t\|, \end{aligned} \tag{5}$$

which is a linear optimization problem and the solution is:

$$d^t = \frac{1}{2\|g_u^t\|^2 \mu} \left(G^T U \Sigma^+ V^T G g_u^t - g_u^t \right), \tag{6}$$

where μ is a related scalar that can make $\|d^t\| = \|g_u^t\|$, i.e., $\mu = \|G^T U \Sigma^+ V^T G g_u^t - g_u^t\| / (2\|g_u^t\|^4)$. The matrices U, Σ, V^T are the singular value decomposition of $GG^T \in \mathbb{R}^{(m-1)\times(m-1)}$, which are not time-consuming to obtain. Σ^+ is the Moore-Penrose pseudoinverse of Σ, i.e., $\Sigma^+ = \text{diag}(\frac{1}{s_1}, \frac{1}{s_2}, \cdots, \frac{1}{s_r}, 0, \cdots, 0)$, where $s_1, s_2, \cdots, s_r$ are the non-zero singular values of GG^T. The detailed proof of Eq.(6) is presented in Appendix.A.1, where we also report the actual computation time of FedOSD. The obtained d^t is closest to $-g_u^t$ and satisfies $Gd^t = 0$, so that it can accelerate unlearning and mitigate the model utility reduction.

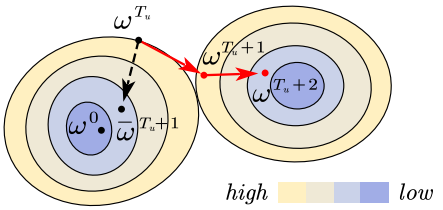

Figure 4: A demo depicting the model reverting issue in post-training. The contour map denotes the local loss of the model on a remaining client. ω^0 is the original model before unlearning. ω^{T_u} is the model after unlearning. The dashed arrow depicts the path of the model update in post-training, where ω^{T_u} moves to $\bar{\omega}^{T_u+1}$ and is closer to ω^0. The red arrows indicate a better path obtained by FedOSD.

3.3 Gradient Projection in Post-training

After unlearning, the target client u leaves the FL system, and the remaining clients undertake a few rounds of FL training to recover the model utility. This phase is referred to as the "post-training" stage (Halimi et al. 2022; Zhao et al. 2023). However, we observe that not only is the model performance across remaining clients recovered, but unexpectedly, the performance on the target client u also improves. It looks like the model remembers what has been forgotten.

One possible case is that the data from the target client u share a similar distribution with the remaining clients' data. Hence, with the model utility being recovered, the model can generalize to client u's data, thereby enhancing the model performance on client u. In general, this issue does not require intervention, because it can even happen on a retrained model without the participation of the target client u.

However, we observe that there is another case called model reverting that requires intervention. As seen in Fig. 4, with the model utility being reduced, the local loss of the remaining clients increased after unlearning. Besides, many previous FU algorithms do not significantly deviate the model from the original model ω^0 during unlearning. Subsequently, when starting post-training, the local gradient g_i^t does not conflict with g_a^t ($g_i^t \cdot g_a^t > 0$), where g_a^t is defined by $g_a^t = \nabla_{\omega^t} \frac{1}{2}\|\omega^t - \omega^0\|^2$. Therefore, the model is driven back to the old local optimal region where ω^0 also resides, so that the model directly recovers what has been forgotten. The experimental results of Table 1 and Fig. 6 substantiate this observation, showing a decreased distance between the model and ω^0 during post-training.

To address this issue, when $g_i^t \cdot g_a^t > 0$, we project the local gradient g_i^t to the normal plane of g_a^t:

$$g'^t_i = g_i^t - \frac{g_i^t \cdot g_a^t}{\|g_a^t\|^2} \cdot g_a^t. \tag{7}$$

Subsequently, each remaining client uploads $g_i'^t$ instead of g_i^t to the server for the aggregation, i.e., $\bar{g}'^t = \frac{1}{|S|}\sum_i g_i'^t$. And the global model is updated by $\omega^{t+1} = \omega^t - \eta\bar{g}'^t$. Given that $g'^t_i \cdot g_a^t = 0, \forall i$, $\bar{g}'^t$ satisfies $\bar{g}'^t \cdot g_a^t = 0$, ensuring that the updated model would not revert towards ω^0. Thus, it addresses the reverting issue in the post-training stage.

4 Experiments

4.1 Experimental Setup

We adopt the model test accuracy on the retained clients (denoted as R-Acc) to evaluate the model utility. To assess the effectiveness of unlearning, we follow (Halimi et al. 2022; Li et al. 2023; Zhao et al. 2023) to implant backdoor triggers into the model by poisoning the target client's training data and flipping the labels (more details can be seen in Appendix.B.1). As a result, the global model becomes vulnerable to the backdoor trigger. The accuracy of the model on these data measures the attack success rate (denoted as ASR), and the low ASR indicates the effective unlearning performance of the algorithm.

Baselines and Hyper-parameters. We first consider the retraining from scratch (denoted as Retraining) and FedEraser (Liu et al. 2021), which is also a kind of retraining but leverages the norms of the local updates stored in the preceding FL training to accelerate retraining. We then encompass well-known FU algorithms including FedRecovery (Zhang et al. 2023), MoDe (Zhao et al. 2023), and the gradient-ascent-based FU methods: EWCSGA (Wu et al. 2022) and FUPGA (Halimi et al. 2022). We follow the settings of (Halimi et al. 2022; Zhang et al. 2023) that all clients utilize Stochastic Gradient Descent (SGD) on local datasets with local epoch $E = 1$. We set the batch size as 200 and the learning rate $\eta \in \{0.005, 0.025, 0.001, 0.0005\}$ decay of 0.999 per round, where the best performance of each method is chosen in comparison. Prior to unlearning, we run FedAvg (McMahan et al. 2017) for 2000 communication rounds to generate the original model ω^0 for unlearning. The max unlearning round is 100, while the max total communication round (including unlearning and post-training) is 200.

Datasets and Models. We follow (Zhao et al. 2023) to evaluate the algorithm performance on the public datasets MNIST (LeCun et al. 1998), FMNIST (Xiao, Rasul, and Vollgraf 2017), and CIFAR-10/100 (Krizhevsky and Hinton 2009), where the training/testing data have already been split. To evaluate the effectiveness of unlearning across varying heterogeneous local data distributions, we consider four scenarios to assign data for clients: (1) Pat-20: We follow (McMahan et al. 2017) to build a pathological non-IID scenario where each client owns the data of 20% classes. For example, in a dataset like MNIST with 10 classes, each client has two classes of the data. (2) Pat-50: It constructs a scenario where each client has 50% classes. (3) Pat-10: It's an extreme data-island scenario where each client has 10% of distinct classes. (4) IID: The data are randomly and equally separated among all clients. We utilize LeNet-5 (LeCun et al. 1998) for MNIST, Multilayer perception (MLP) (Popescu et al. 2009) for FMNIST, CNN (Halimi et al. 2022) with two convolution layers for CIFAR-10, and NFResNet-18 (Brock, De, and Smith 2021) for CIFAR-100.

4.2 Evaluation of Unlearning and Model Utility

We first evaluate the ASR and R-Acc of the model at the end of both the unlearning stage and post-training stage on FMNIST and CIFAR-10. One of the ten clients is randomly selected as the target client requesting for unlearn-

	FMNIST						CIFAR-10					
	Pat-20		Pat-50		IID		Pat-20		Pat-50		IID	
Algorithm	ASR	R-Acc	ASR	R-Acc	ASR	R-Acc	ASR	R-Acc	ASR	R-Acc	ASR	R-Acc
ω^0	.991	.852(.113)	.957	.869(.013)	.893	.898(.010)	.897	.589(.115)	.754	.658(.016)	.243	.731(.013)
Retraining	.004	.760(.228)	.002	.817(.025)	.002	.840(.015)	.047	.507(.106)	.009	.583(.149)	.022	.511(.013)
FedEraser	.005	.763(.171)	.011	.810(.102)	.002	.872(.009)	.098	.454(.158)	.026	.571(.128)	.016	.683(.012)
FedRecovery[1]	.637	.761(.279)	.693	.823(.092)	.498	.871(.012)	.156	.454(.337)	.102	.476(.346)	.015	.692(.017)
MoDe[1]	.003	.667(.246)	.005	.777(.046)	.002	.792(.012)	.145	.256(.162)	.066	.199(.119)	.025	.481(.018)
EWCSGA[1]	.000	.255(.259)	.000	.233(.261)	.101	.126(.009)	.000	.199(.372)	.000	.381(.426)	.018	.259(.010)
FUPGA[1]	.000	.227(.254)	.000	.178(.200)	.101	.105(.008)	.000	.202(.373)	.000	.388(.433)	.019	.271(.013)
FedOSD[1]	**.000**	**.757(.187)**	**.000**	**.806(.042)**	**.000**	**.884(.011)**	**.000**	**.549(.185)**	**.000**	**.602(.175)**	**.000**	**.696(.016)**
FedRecovery[2]	.960^r	.857(.112)	.873^r	.876(.013)	.806^r	.898(.011)	.785^r	.607(.119)	.598^r	.643(.138)	.155^r	.737(.016)
MoDe[2]	.007	.744(.252)	.003	.816(.028)	.002	.843(.014)	.060	.519(.117)	.035	.582(.173)	.016	.703(.016)
EWCSGA[2]	.935^r	.836(.173)	.400^r	.869(.013)	.378^r	.896(.012)	.581^r	.591(.194)	.592^r	.652(.118)	.140^r	.736(.016)
FUPGA[2]	.857^r	.837(.185)	.745^r	.875(.013)	.199^r	.894(.009)	.662 ^{r}	.599(.157)	.602^r	.658(.091)	.144^r	.737(.014)
FedOSD[2]	.023	.851(.105)	.021	.874(.014)	.004	.897(.011)	.027	.606(.101)	.016	.659(.017)	.030	.734(.015)

Table 1: The ASR, the mean R-Acc (and the std.) of the model. The row of ω^0 denotes the initial state before unlearning. The '1' marked following the algorithm name represents the results after unlearning, while '2' denotes the results after post-training. The signal 'r' in the columns of ASR ignifies an increase of the ASR value because of the model reverting during post-training.

	MNIST			CIFAR-100		
Algorithm	ASR	R-Acc	NC	ASR	R-Acc	NC
ω^0	.997	.963	-	.584	.394	-
FedRecovery	.038	.716	3.00	.000	.214	1.00
MoDe	.039	.723	3.47	.027	.160	3.13
EWCSGA	.000	.527	7.45	.000	.093	7.95
FUPGA	.000	.535	7.38	.000	.090	7.92
FedOSD	**.000**	**.924**	**0.00**	**.000**	**.369**	**0.00**

Table 2: ASR, R-Acc, and NC, the mean number of retained clients per round whose gradients conflict with the model update direction in Pat-20 on MNIST and CIFAR-100.

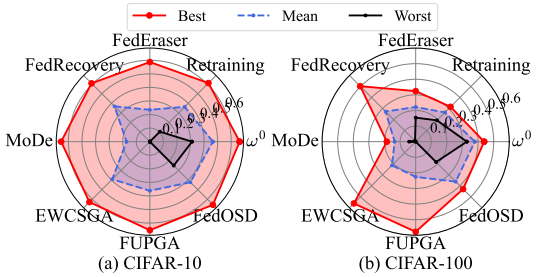

Figure 5: The best, the average, and the worst R-Acc across clients in Pat-10 on (a) CIFAR-10 and (b) CIFAR-100.

ing. Table. 1 lists the comparison results. It can be seen that in the unlearning stage, the gradient-ascent-based FU algorithms such as EWCSGA and FUPGA achieve more complete unlearning in non-IID scenarios, evidenced by their ASR reaching 0, but they experience a more pronounced reduction in R-Acc. Benefiting from the UCE loss and the orthogonal steepest descent update direction, the proposed FedOSD can successfully unlearn the target client data while suffering less utility reduction than others. Besides, since FedRecovery performs unlearning relies solely on the prestored historical FL training information, it cannot guarantee the unlearning effect in all scenarios. During the post-training stage, FedRecovery, EWCSGA, and FUPGA can recover the R-Acc to a level comparable to or exceeding that of the initial state. However, their models gravitate towards the initial ω^0, leading to the models remembering what has been erased, and thus the ASR values rise significantly. In comparison, FedOSD can recover the model utility without suffering the model reverting issue. More experimental results on MNIST and CIFAR-100 are available in Appendix.B.2.

To elucidate the negative impact of gradient conflicts on the model utility during unlearning, we report ASR, R-Acc, and the average number of retained clients experiencing gradient conflicts with the model update direction d^t in Table. 2. The results demonstrate that mitigating the conflict between d^t and the remaining clients' gradients can significantly alleviate reductions in the model utility.

Besides, Fig. 5 depicts the results in Pat-10 to evaluate the effect of unlearning on the model utility when some classes of data are completely removed. Compared with FedOSD, the model utility reduction on previous FU methods is considerably unfair, where the R-acc values are even approaching 0. In contrast, FedOSD more effectively maintains the model's performance on the remaining clients.

Moreover, we visualize the curves of ASR, R-Acc, and the distance between ω^t and ω^0 during unlearning and post-training in Fig. 6. Notably, the unlearning stage of FedRecovery only comprises a single round, as it performs unlearning relying solely on the historical information of the previous FL training. The results demonstrate that FedOSD successfully achieves a zero ASR while maintaining the highest model utility during unlearning. The distance curve in the post-training stage verifies that the models of FedRecovery, EWCSGA, and FUPGA tend to revert towards ω^0, evidenced by the decreasing distance, thereby leading

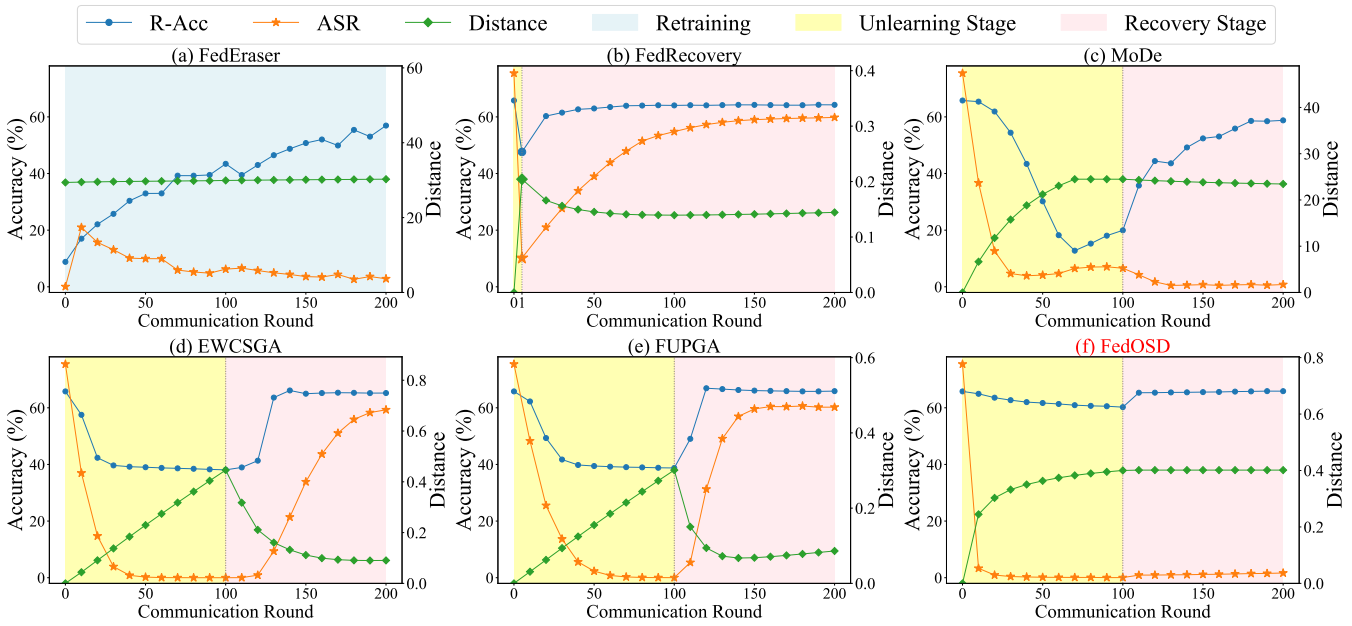

Figure 6: The ASR, the mean R-Acc, and the distance away from ω^0 during unlearning and post-training in Pat-5 on CIFAR-10.

	FMNIST		CIFAR-10		CIFAR-100	
Method	ASR	R-Acc	ASR	R-Acc	ASR	R-Acc
ω^0	.957	.869	.754	.658	.433	.437
FedOSD^1	**.000**	.806	**.000**	.602	**.000**	.399
M1^1	.000	.163	.000	.224	.000	.014
M2^1	.000	.317	.000	.391	.000	.267
M3^1	.835	**.886**	.331	**.697**	.159	**.476**
M4^1	.046	.693	.009	.566	.004	.360
M5^1	.000	.806	.000	.602	.000	.399
FedOSD^2	**.021**	.874	**.024**	.659	**.056**	.458
M5^2	.244^r	.878	.340^r	.655	.138^r	.462

Table 3: The ASR, the mean R-Acc of the model in the ablation studies. '1' marks the unlearning stage and '2' denotes post-training. 'r' means suffering the model reverting issue.

to an increase in ASR, which suggests a recovery of previously unlearned information. In contrast, FedOSD prevents the model from moving back, thereby ensuring the recovery of model utility without suffering the model reverting issue during post-training.

4.3 Ablation Experiments

In Table. 3, we evaluate the performance of several variants of FedOSD (M1 to M5) to study the effect of each part.

M1: Do not use the UCE loss. Instead, the target client utilizes Gradient Ascent on the CE loss to unlearn. The results demonstrate that GA would destroy the model utility.

M2: Replace the orthogonal steepest descent direction d^t to $-g_u^t$ for updating the unlearning model, which would conflict with retained clients' gradients. As a result, the model utility suffers more reduction than FedOSD.

M3: During unlearning, using Multiple Gradient Descent algorithm (Fliege and Svaiter 2000; Pan et al. 2024) to obtain a common descent direction d^t that satisfies $d^t \cdot g_i^t < 0, \forall i \neq u$, which can both reduce the UCE loss of the target client and the CE loss of remaining clients in unlearning. The results depict that while this strategy does not compromise model utility, it fails to achieve the unlearning goal, verifying the analysis in Section 3.2.

M4: Randomly select a solution d^t from the solutions to $G \cdot d^t = \vec{0}$ that also satisfies $d^t \cdot g_u^t < 0$ to update the model for unlearning. Since the obtained d^t would deviate a lot from $-g_u^t$, the result of ASR is higher than that of FedOSD. If we tune a larger learning rate to enhance the unlearning performance, it would further harm the model utility.

M5: Remove the gradient projection strategy in the post-training stage. It results in the model reverting issue, with a significant increase in ASR, verifying it's necessary to prevent the model from moving back to ω^0 during post-training.

5 Conclusion and Future Work

In this work, we identify the convergence issue of Gradient Ascent and demonstrate the necessity of mitigating the gradient conflict in Federated Unlearning. Moreover, we highlight the issue of model reverting during post-training, which adversely affects the unlearning performance. To address these issues, we propose FedOSD, which modifies the Cross-Entropy loss to an unlearning version and achieves an orthogonal steepest descent model direction for unlearning. Extensive experiments verify that FedOSD outperforms SOTA FU methods in terms of the unlearning effect and mitigating the model utility reduction. A number of interesting topics warrant future exploration, including the design of the unlearning version of other loss functions such as MSE loss, and further enhancing fairness and privacy protection in FU.

References

Bourtoule, L.; Chandrasekaran, V.; Choquette-Choo, C. A.; Jia, H.; Travers, A.; Zhang, B.; Lie, D.; and Papernot, N. 2021. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*, 141–159. IEEE.

Brock, A.; De, S.; and Smith, S. L. 2021. Characterizing signal propagation to close the performance gap in unnormalized ResNets. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

De, K.; and Pedersen, M. 2021. Impact of colour on robustness of deep neural networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, 21–30.

Fliege, J.; and Svaiter, B. F. 2000. Steepest descent methods for multicriteria optimization. *Mathematical Methods of Operations Research*, 51(3): 479–494.

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

Harding, E. L.; Vanto, J. J.; Clark, R.; Hannah Ji, L.; and Ainsworth, S. C. 2019. Understanding the scope and impact of the california consumer privacy act of 2018. *Journal of Data Protection & Privacy*, 2(3): 234–253.

Krizhevsky, A.; and Hinton, G. 2009. Learning multiple layers of features from tiny images. *Handbook of Systemic Autoimmune Diseases*, 1(4).

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324.

Li, G.; Shen, L.; Sun, Y.; Hu, Y.; Hu, H.; and Tao, D. 2023. Subspace based federated unlearning. *arXiv preprint arXiv:2302.12448*.

Li, T.; Sahu, A. K.; Talwalkar, A.; and Smith, V. 2020. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3): 50–60.

Liu, G.; Ma, X.; Yang, Y.; Wang, C.; and Liu, J. 2021. Federaser: Enabling efficient client-level data removal from federated learning models. In *2021 IEEE/ACM 29th international symposium on quality of service (IWQOS)*, 1–10. IEEE.

Liu, Z.; Jiang, Y.; Shen, J.; Peng, M.; Lam, K.-Y.; and Yuan, X. 2023. A survey on federated unlearning: Challenges, methods, and future directions. *arXiv preprint arXiv:2310.20448*.

McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, 1273–1282. PMLR.

Pan, Z.; Li, C.; Yu, F.; Wang, S.; Wang, H.; Tang, X.; and Zhao, J. 2024. FedLF: Layer-Wise Fair Federated Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 14527–14535.

Pan, Z.; Wang, S.; Li, C.; Wang, H.; Tang, X.; and Zhao, J. 2023. Fedmdfg: Federated learning with multi-gradient descent and fair guidance. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 9364–9371.

Popescu, M.-C.; Balas, V. E.; Perescu-Popescu, L.; and Mastorakis, N. 2009. Multilayer perceptron and neural networks. *WSEAS Transactions on Circuits and Systems*, 8(7): 579–588.

Su, N.; and Li, B. 2023. Asynchronous federated unlearning. In *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*, 1–10. IEEE.

Voigt, P.; and Von Bussche, A. 2017. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 10(3152676): 10–5555.

Wang, Z.; Fan, X.; Qi, J.; Wen, C.; Wang, C.; and Yu, R. 2021. Federated Learning with Fair Averaging. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, 1615–1623. IJCAI Organization.

Wu, C.; Zhu, S.; and Mitra, P. 2022. Federated unlearning with knowledge distillation. *arXiv preprint arXiv:2201.09441*.

Wu, L.; Guo, S.; Wang, J.; Hong, Z.; Zhang, J.; and Ding, Y. 2022. Federated unlearning: Guarantee the right of clients to forget. *IEEE Network*, 36(5): 129–135.

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Yang, J.; and Zhao, Y. 2023. A survey of federated unlearning: A taxonomy, challenges and future directions. *arXiv preprint arXiv:2310.19218*.

Ye, G.; Chen, T.; Hung Nguyen, Q. V.; and Yin, H. 2024. Heterogeneous decentralised machine unlearning with seed model distillation. *CAAI Transactions on Intelligence Technology*.

Yu, B.; Mao, W.; Lv, Y.; Zhang, C.; and Xie, Y. 2022. A survey on federated learning in data mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 12(1): e1443.

Zhang, L.; Zhu, T.; Zhang, H.; Xiong, P.; and Zhou, W. 2023. Fedrecovery: Differentially private machine unlearning for federated learning frameworks. *IEEE Transactions on Information Forensics and Security*.

Zhao, Y.; Wang, P.; Qi, H.; Huang, J.; Wei, Z.; and Zhang, Q. 2023. Federated unlearning with momentum degradation. *IEEE Internet of Things Journal*.

Reproducibility Checklist

1. This paper:
   (a) Includes a conceptual outline and/or pseudocode description of AI methods introduced. (Yes)
   (b) Clearly delineates statements that are opinions, hypotheses, and speculation from objective facts and results. (Yes)
   (c) Provides well-marked pedagogical references for less-familiar readers to gain background necessary to replicate the paper. (Yes)

2. Does this paper make theoretical contributions? (Yes)
   (a) All assumptions and restrictions are stated clearly and formally. (Yes)
   (b) All novel claims are stated formally (e.g., in theorem statements). (Yes)
   (c) Proofs of all novel claims are included. (Yes)
   (d) Proof sketches or intuitions are given for complex and/or novel results. (Yes)
   (e) Appropriate citations to theoretical tools used are given. (Yes)
   (f) All theoretical claims are demonstrated empirically to hold. (Yes)
   (g) All experimental code used to eliminate or disprove claims is included. (Yes)
3. Does this paper rely on one or more datasets? (Yes)
   (a) A motivation is given for why the experiments are conducted on the selected datasets. (Yes)
   (b) All novel datasets introduced in this paper are included in a data appendix. (N/A)
   (c) All novel datasets introduced in this paper will be made publicly available upon publication of the paper with a license that allows free usage for research purposes. (N/A)
   (d) All datasets drawn from the existing literature (potentially including authors' own previously published work) are accompanied by appropriate citations. (Yes)
   (e) All datasets drawn from the existing literature (potentially including authors' own previously published work) are publicly available. (Yes)
   (f) All datasets that are not publicly available are described in detail, with an explanation of why publicly available alternatives are not scientifically satisfying. (N/A)
4. Does this paper include computational experiments? (Yes)
   (a) Any code required for pre-processing data is included in the appendix. (Yes)
   (b) All source code required for conducting and analyzing the experiments is included in a code appendix. (Yes)
   (c) All source code required for conducting and analyzing the experiments will be made publicly available upon publication of the paper with a license that allows free usage for research purposes. (Yes)
   (d) All source code implementing new methods has comments detailing the implementation, with references to the paper where each step comes from. (Yes)
   (e) If an algorithm depends on randomness, then the method used for setting seeds is described in a way sufficient to allow replication of results. (Yes)
   (f) This paper specifies the computing infrastructure used for running experiments (hardware and software), including GPU/CPU models, amount of memory, operating system, names, and versions of relevant software libraries and frameworks. (Yes)
   (g) This paper formally describes evaluation metrics used and explains the motivation for choosing these metrics. (Yes)
   (h) This paper states the number of algorithm runs used to compute each reported result. (Yes)
   (i) Analysis of experiments goes beyond single-dimensional summaries of performance (e.g., average; median) to include measures of variation, confidence, or other distributional information. (Yes)
   (j) The significance of any improvement or decrease in performance is judged using appropriate statistical tests (e.g., Wilcoxon signed-rank). (Yes)
   (k) This paper lists all final (hyper-)parameters used for each model/algorithm in the paper's experiments. (Yes)
   (l) This paper states the number and range of values tried per (hyper-)parameter during the development of the paper, along with the criteria used for selecting the final parameter setting. (Yes)