Lockdown: Backdoor Defense for Federated Learning with Isolated Subspace Training

Anonymous Authors[1]

Abstract

Federated learning (FL) is known to be vulnerable to backdoor attacks due to its distributed computing nature. We in this paper propose a backdoor defense solution, dubbed Lockdown to counter the backdoor attack launched by malicious clients. Lockdown isolates the training subspaces for different clients, identifies the malicious/dummy parameters within the global model, and subsequently purges them to cure the model. Empirical results show that Lockdown achieves *superior* and *consistent* defense performance without sacrificing much on the model's accuracy on the benign samples. In addition, Lockdown enjoys extra benefits, i.e., *communication and model complexity reduction*, which are all desirable properties in FL. Our code is available at https://github.com/LockdownAuthor/Lockdown.

1. Introduction

Federated Learning (FL) (McMahan et al., 2016) is a privacy-preserving machine learning paradigm that allows the training surrogates (i.e., clients) to collectively train a global model with the data resided in local devices. However, because the training data and potentially the training process on the clients lacks valid supervision, it is possible that attackers can launch data poisoning attack on the global model (Tolpegin et al., 2020), so as to manipulate the prediction of the model to reach goal of their interest.

Backdoor attack is one of those kinds of data poisoning attack, which is stealthy, and is also disruptive to the normal function of the model. Specifically, the prediction of the model can be manipulated such that it will consistently predict one (or some) specific target label whenever it is given samples with a backdoor trigger. See Figure 1 for an illustration of how to conduct backdoor attack on FL. Backdoor attack poses serious threats to many security-critical applications, e.g., biometric authentication and autonomous driving (Chow & Liu, 2021), which vitiates the value of large-scale deployment of FL. Therefore, an effective defense solution that can mitigate such risk in FL is in an urgent need.

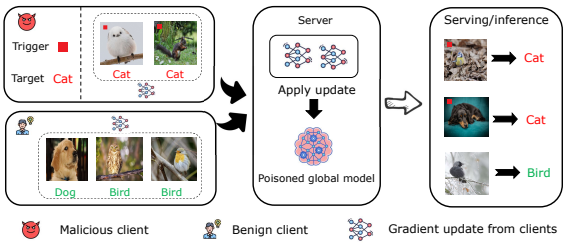

Figure 1. Illustration of backdoor attack on FL. Firstly, clients train on their dataset to obtain gradient update, which are sent to server. Secondly, the malicious gradient update is aggregated into the global model by server without awareness. Finally, the global model output is manipulated towards those samples with a backdoor trigger, which we say the global model is "poisoned".

The main stream of existing research to defend backdoor attack in FL can be mainly classified into three genres, i) outlier detection (Tolpegin et al., 2020), (Tahmasebian et al., 2022), (Ozdayi et al., 2021). ii) certified robustness (Xie et al., 2021), (Alfarra et al., 2022), and iii) adversarial training (Zizzo et al., 2020; Shah et al., 2021). Though existing defenses can mitigate attack to some extent, they are still far from their maturity. In summary, we highlight the following common weaknesses of state-of-the-art solutions.

Existing defenses do not enjoy consistent efficacy in a wide variety of attack settings. Particularly, the outlier detection-based technique, may not be reliable when the training data is Non-IID. The gradient update from a benign client may easily be recognized as an outlier, given that gradient updates from different clients are intrinsically heterogeneous, and vice versa, a malicious update can also be recognized as benign and aggregated to the global model.

Existing defenses require extra computation in either the training or inference phases. For example, randomized smoothing, the key component for certified robustness defense, requires multiple forward pass of data in the inference phase for producing one effective prediction. Analogously,

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

adversarial training requires to generate adversarial samples, and demands extra data pass to train such samples.

Motivated by the existing gap, we in this paper explore a new genre of defense via connecting backdoor defense with isolated subspace training. Our high-level idea is to isolate the poison effect into a subspace of the whole parameter space, and subsequently prune out those corrupted parameters in order to diminish the function for classifying backdoor features. All in all, we summarize the following questions that motivate our research:

1. *How to isolate the poisoned parameters within a particular subspace such that it does not contaminate the whole model's parameter space?*
2. *How to identify and purge the poisoned parameters such that the model still functions normally when given benign inputs, but its backdoor function is perturbed?*

Driven by the course of exploration, we propose Lockdown, a solution that utilizes isolated subspace training to conduct backdoor defense. Our experiments demonstrate that: i) Lockdown reduces the Attack Success Ratio (ASR) by up-to 90% compared to FedAvg without defense. ii) Lockdown consistently performs better than SOTA defense solutions in various attack settings. Specifically, Lockdown acquires 93% of ASR reduction with only 5% drop of benign accuracy when the data distribution is Non-IID (the other two defense baselines RLR and Krum respectively acquire 14.2% and 86.1% ASR reduction while sacrificing 16.2% and 43.6% benign accuracy in the same setting). iii) Lockdown reduces both downlink and uplink communication by at least 0.5x, and the number of parameters used in the model training and inference phases are also accordingly reduced by at least 0.5x thanks to the removal of malicious/dummy parameters.

To the end, we summarize our contribution as follows:

- We propose Lockdown, a backdoor defense solution that utilizes the idea of isolated subspace training. As a bonus, Lockdown i) enjoys communication reduction between server and clients, and ii) lowers the model's training/inference complexity since only a subspace of the model is trained and deployed.
- We conduct empirical evaluations to demonstrate the effectiveness of Lockdown. Experimental results demonstrate that Lockdown *consistently* outperforms existing defense baselines under different attack settings (attack method, attacker number and poison ratio) and different data distributions (IID and Non-IID).
- Ablation study and hyper-parameter sensitivity analysis are conducted to verify the individual functionality of each component of Lockdown.

2. Related Work

Federated Learning. Federated learning (McMahan et al., 2016) is a privacy-preserving distributed training paradigm that allows clients to collectively train a global model from distributed training data. Recent works on FL mostly lie in the optimization aspect (Karimireddy et al., 2020; Li et al., 2018; Acar et al., 2021; Zhang et al., 2020), which study how to mitigate the Non-IID issue (Zhao et al., 2018).

Federated backdoor attack and defense. Classical backdoor attack in FL is empirically proven to be effective in (Bagdasaryan et al., 2020), and subsequently, new types of attack are developed, e.g., edge-case backdoor (Wang et al., 2020), stealthy model poisoning (Bhagoji et al., 2019) and Distributed Backdoor Attack (DBA) (Xie et al., 2019).

Backdoor defense solutions are developed to counter the potential threat posed by the backdoor attack, which can be classified into three main genres. The first genre is outlier detection. (Tolpegin et al., 2020) use PCA to decompose the gradient updates from clients, and exclude those clients that frequently upload outlier updates from training. Similar ideas to analyze the gradient update from clients are also adopted in (Tahmasebian et al., 2022), which utilizes truth inference to estimate client reliability, and (Ozdayi et al., 2021), which utilizes inverse learning rate towards the outlier gradient update. However, data heterogeneity will significantly vitiate the practical performance of this genre of methods. The second genre is certified robustness. Certified robustness relies on randomized smoothing(Cohen et al., 2019), an approach with theoretical certification of the model robustness. Subsequent techniques, e.g., weight smoothing (Xie et al., 2021) and group ensemble (Cao et al., 2022) are studied for the defense of federated backdoor. The last genre is adversarial training (Zizzo et al., 2020; Shah et al., 2021), which generates adversarial samples to improve the model's robustness. However, both certified robustness and adversarial training require more computation in either the training or deployment stage.

Sparse training. Sparse training is originally designed to reduce the model complexity for both the training and deployment stage. SET (Mocanu et al., 2018) first proposes the idea of dynamic subspace searching with alternative pruning and recovery process, and are empirically studied with the concept of in-time over-parameterization (Liu et al., 2021b;a). (Evci et al., 2020) further introduces gradient information to guide the mask searching process. This model compression technique has recently been extended to FL (Bibikar et al., 2022; Huang et al., 2022).

We in this paper utilize sparse training technique to develop a new genre of backdoor defense solution. To our best knowledge, this is the first attempt that connects sparse training with backdoor defense in FL.

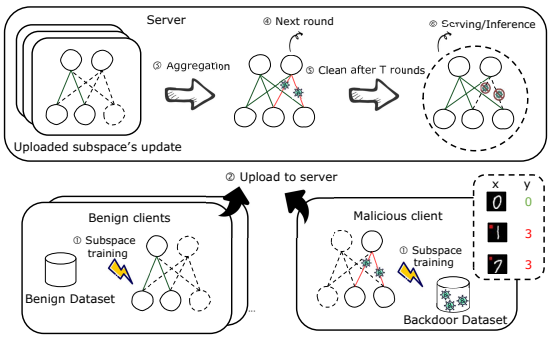

Figure 2. Overview of Lockdown. Firstly, clients apply isolated subspace training over the global model using their own (potentially poisoned) dataset. Secondly, clients upload gradient update to the server. Thirdly, the server aggregates the update, producing the next global model. After T rounds of training, Lockdown performs network cleaning with consensus fusion (see Eq. (6)).

3. Threat models

In this section we present threat models that formalize backdoor attack on FL. We consider three threat models, named *weak*, *medium* and *strong*. All the models allow multiple attackers co-exist in the system. We use N to denote the number of attackers out of M total clients.

Table 1. Permission of threat models.

Permission \Threat model	weak	medium	strong
Data manipulation	✓	✓	✓
Algorithm manipulation	✗	✓	✓
Global information Access	✗	✗	✓
Aggregation manipulation	✗	✗	✗

Permission. Different threats models are given different control permissions. Specifically, *Weak* model allows the attackers to arbitrarily manipulate its local data, but cannot control its local training process, while attackers in *medium* and *strong* model can control the local training process. Only *strong* model allow the attackers to obtain other benign client's information, probably by intercepting their communication with the server. For control permission in the weak threat model, technique like trusted execution environment (Mo et al., 2021) can be applied in order control each client run the designated server program, and thereby disabling algorithm manipulation. Existing literature usually adopt medium threat model, in which attackers can do whatever they like in their local devices, but has not extra information from server or from other benign clients. The permission of the threat models are summarized in Table 1.

Malicious objective. Denote the set of benign clients as $\mathcal{M}$ and the set of malicious clients as $\mathcal{N}$. Formally, we characterize the objective of malicious clients as follows:

$$\min_{\boldsymbol{w}} \frac{1}{M}\left(\sum_{i\in\mathcal{N}} \tilde{f}_i(\boldsymbol{w}) + \sum_{i\in\mathcal{M}/\mathcal{N}} f_i(\boldsymbol{w})\right) \tag{1}$$

where $\tilde{f}_i(\boldsymbol{w}) \triangleq \frac{1}{|\tilde{\mathcal{D}}_i|}\sum_{(\boldsymbol{x}_j,y_j)\in\tilde{\mathcal{D}}_i} CE(\boldsymbol{w};\boldsymbol{x}_j,y_j))$ is the malicious objective of one specific client, $f_i(\boldsymbol{w}) \triangleq \frac{1}{|\mathcal{D}_i|}\sum_{(\boldsymbol{x}_j,y_j)\in\mathcal{D}_i} CE(\boldsymbol{w};\boldsymbol{x}_j,y_j))$ is the benign objective, and $\tilde{\mathcal{D}}_i$ and $\mathcal{D}_i$ are respectively the backdoor dataset and pristine dataset.

Attack methods. Different attack methods is eligible in different threat models. For weak threat model. only data-level backdoor, e.g., BadNet (Gu et al., 2017), DBA (Xie et al., 2019), and Sinusoidal (Barni et al., 2019) can be applied, since the attacker has not control to the local training, but only has access to the data for backdooring. For medium threat model, the attacker can perform a wider range of attack methods, e.g., Scaling (Bagdasaryan et al., 2020), FixMask, Snooper, etc, which requires the modification of the local training process. For strong threat model, we test Neurontoxin (Zhang et al., 2022), and Omniscient, which requires extra server-side information to aid the attack. We postpone further analysis to the experiment section.

Defense goal. While solving the benign objective (see FL general objective in (McMahan et al., 2016)), we expect the global model $\boldsymbol{w}$ to be able to resist backdoor attack. In other words, the global model should minimize the benign objective while maximizing the malicious objective.

4. Methodology

In this section, we present Lockdown to counter the backdoor attack. The workflow of Lockdown is summarized in Algorithm 1 and Figure 2. The auxiliary functions used in the main algorithm are available in Algorithm 2.

Our high-level idea to defend backdoor is as follows. Firstly, we employ *isolated subspace training* to constraint the training of each client into its own subspace, such that the backdoor data cannot contaminate all the parameters. Then we employ *dynamic subspace searching* for the clients to search for local subspaces using their local datasets, which involve parameters that they deem important. After the above local procedures, the server will *aggregate* the gradient updates into the corresponding subspaces of global model, and continues the next round of training. Repeating T rounds of training, the server can identify the malicious parameters by *consensus fusion* under the hypothesis that the malicious/dummy parameters should appear less among all the subspaces, which are then pruned to mitigate backdoor function. In the following, we detail each step of Lockdown.

Subspace initialization. We use binary masks to identify the training subspace of each client formally. For each client, they share the same mask when initialization. We follow (Evci et al., 2020) to use ERK for random mask initialization, in which different layers of the model are designated different sparsity (See Appendix A for details).

Algorithm 1 Main Loop of Lockdown

input Training iteration T; Local steps K; Learning rate η; Random seed $seed$; Shrinkage penalty λ;
output Clean model for deployment $\tilde{\boldsymbol{w}}_T$

main Server's Main Loop
  Randomly initialize global model $\boldsymbol{w}_0$
  $\boldsymbol{m}_{i,0}$ = **SubspaceInit**($seed$) for $i \in \mathcal{M}$
  for $t = 0, 1, \ldots, T-1$ **do**
    for $i \in \mathcal{M}$ **do**
      Send $\tilde{\boldsymbol{w}}_{i,t,0} = \boldsymbol{m}_{i,t} \odot \boldsymbol{w}_t$ to client k
      Call Client i's main loop for training
      Received $\boldsymbol{U}_{i,t}$ and $\boldsymbol{m}_{i,t+1}$
    end for
    /* Subspaces average with mask */
    $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \frac{1}{\sum_{i\in\mathcal{M}} \boldsymbol{m}_{i,t}} \sum_{i\in\mathcal{M}} \boldsymbol{U}_{i,t}$
  end for
  /*Poison effect erasing with consensus fusion*/
  $\tilde{\boldsymbol{w}}_T$ = **ConsensusFusion**$(\boldsymbol{w}_T, \boldsymbol{m}_{1,T}, \ldots, \boldsymbol{m}_{M,T})$
  Deploy $\tilde{\boldsymbol{w}}_T$ for serving/inference.
end main

main Client's Main Loop
  /* Subspace training with shrinkage*/
  for $k = 0, 1, \ldots, K-1$ **do**
    $\boldsymbol{w}_{i,t,k+1} = \mathcal{S}_\lambda(\boldsymbol{w}_{i,t,k} - \eta \boldsymbol{m}_{i,t} \odot \nabla f_i(\boldsymbol{w}_{i,t,k}; \xi))$
  end for
  $\boldsymbol{U}_{i,t} = \tilde{\boldsymbol{w}}_{i,t,0} - \tilde{\boldsymbol{w}}_{i,t,K}$
  $\boldsymbol{m}_{i,t+1}$ = **SubspaceSearch**$(\boldsymbol{w}_{i,t,K}, \boldsymbol{m}_{i,t})$
  Send $\boldsymbol{U}_{i,t}$ and $\boldsymbol{m}_{i,t+1}$ to server
end main

Algorithm 2 Auxiliary Functions for Lockdown

input Initial pruning rate α_0; overall sparsity s; consensus fusion threshold θ; Layers of model L;

function SubSpaceInit(seed)
  Init layer sparsity $\{s_l\}$ given overall sparsity s with ERK.
  for $l = 0, 1, \ldots, L-1$ **do**
    Randomly initialize $\boldsymbol{m}_{i,0}^{(l)}$ with sparsity s_l and $seed$
  end for
  Return $\boldsymbol{m}_{i,0}$
end function

function SubspaceSearch$(\boldsymbol{w}_{i,t,K}, \boldsymbol{m}_{i,t})$
  Decay α_t with initial pruning rate α_0
  for $l = 0, 1, \ldots, L-1$ **do**
    Update $\boldsymbol{m}_{i,t+\frac{1}{2}}^{(l)}$ via pruning $a_t\%$ of parameters
  end for
  Input data to obtain gradient $\nabla f_i(\boldsymbol{w}_{i,t,K} \odot \boldsymbol{m}_{i,t+\frac{1}{2}})$
  for $l = 0, 1, \ldots, L-1$ **do**
    Update $\boldsymbol{m}_{i,t+1}^{(l)}$ via recovering $a_t\%$ of parameters with gradient $\nabla f_i^{(l)}(\boldsymbol{w}_{i,t,K} \odot \boldsymbol{m}_{i,t+\frac{1}{2}})$
  end for
  Return $\boldsymbol{m}_{i,t+1}$
end function

function ConsensusFusion$(\boldsymbol{w}_T, \boldsymbol{m}_{1,T}, \ldots, \boldsymbol{m}_{M,T})$
  for $l = 0, 1, \ldots, L-1$ **do**
    $\boldsymbol{m}_g^{(l)} = \mathcal{T}_\theta(\boldsymbol{m}_{1,T}^{(l)}, \ldots, \boldsymbol{m}_{M,T}^{(l)})$
  end for
  Return $\boldsymbol{w}_T \odot \boldsymbol{m}_g$
end function

This initialization is necessary to maintain model's normal function, per our evaluation in the ablation study.

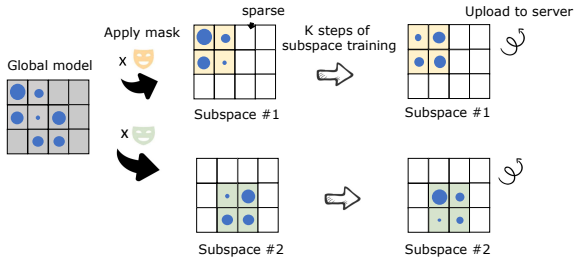

Figure 3. Illustration of isolated subspace training, where the blue circle denotes the magnitude of parameters, and the shaded area denotes the isolated subspace (i.e., the mask). Clients receive a subspace of the global model, and conduct K steps of isolated training within the subspace before updating to the server.

Isolated subspace training. Each client performs multiple local steps to train parameters within its isolated subspace (which is enforced by its mask). Specifically, for local step $k = 0, 1, \ldots, K-1$, clients do the following update:

$$\boldsymbol{w}_{i,t,k+1} = \boldsymbol{w}_{i,t,k} - \eta \boldsymbol{m}_{i,t} \odot \nabla f_i(\boldsymbol{w}_{i,t,k}; \xi) \qquad (2)$$

where ξ is a piece of random sample within the local dataset, and $\odot$ denotes Hadamard product. As shown, the binary mask $\boldsymbol{m}_{i,t}$ is applied to the gradient, which isolates the training into i-th client's own subspace. Isolated training prevents the malicious data from contaminating all the parameters within the global parameter space. The isolated subspace training is illustrated in Figure 3.

Subspace searching. We conduct subspace searching for clients to search for their own subspaces (i.e., the parameters that are deemed important) according to the data they have. Specifically, the subspace searching can be decomposed to two phases (See Figure 4 for visualization).

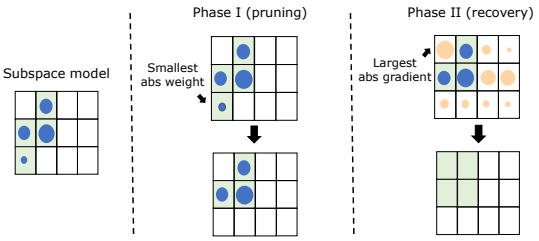

Figure 4. Illustration of subspace searching process. The blue circles denote the magnitude of parameters, and the orange circles denote the magnitude of gradient (which is obtained by one pass of local data). On the first phase, each client prunes the smallest-$\alpha_t\%$ of parameters based on the order of their magnitude. On the second phase, each client recovers the largest-$\alpha_t\%$ of parameters based on the order of gradient magnitude.

1. **Subspace pruning.** In this phase, we prune the unim-

portant parameters within the client's current subspace that do not function at all. Before we introduce how to prune, we first re-write Eq.(2) in order to reveal the unimportant parameters in the subspace, as follows:

$$\boldsymbol{w}_{i,t,k+1} = \mathcal{S}_\lambda(\boldsymbol{w}_{i,t,k} - \eta \boldsymbol{m}_{i,t} \odot \nabla f_i(\boldsymbol{w}_{i,t,k}; \xi)) \quad (3)$$

where operator $\mathcal{S}_\lambda(\cdot)$ is a shrinkage operator as follows.

$$[\mathcal{S}_\lambda(\boldsymbol{x})]_j = \begin{cases} x_j - \lambda & x_j \geq 0 \\ x_j + \lambda & x_j < 0 \end{cases} \quad (4)$$

where λ is the shrinkage penalty. This shrinkage operator ensures that the unimportant parameters within the subspace shrink to 0, while those important parameters are more resilient, therefore, can retain their values.

After K steps of local training, we can readily identify the unimportant parameters via identifying those with the smallest magnitude within the subspace. Then we prune those smallest $\alpha_t\%$ parameters within each layer, and accordingly update the mask to $\boldsymbol{m}_{i,t+\frac{1}{2}}$. We use cosine annealing to decay α_t from α_0. The detailed decay process is postponed to the Appendix A.

2. **Subspace recovery.** After pruning, the client recovers the same amount of parameters to its subspace for exploration. To identify which parameters should be recovered, we use each client's data to extract the gradient w.r.t the weights after pruned, i.e., to extract $\nabla f_i(\boldsymbol{w}_{i,t,K} \odot \boldsymbol{m}_{i,t+\frac{1}{2}})$. Then we recover $\alpha_t\%$ parameters by identifying those with top-$\alpha_t\%$ gradient magnitude within each layer, and accordingly update the mask to $\boldsymbol{m}_{i,t+1}$. The intuition is that for an important parameter over the local data, its gradient magnitude should be larger than the unimportant ones. Therefore, extending the subspace to these parameters can boost the chance of finding important parameters over local data.

Aggregation. Once subspace training is done, the clients upload the gradient updates of the local subspace to the server for aggregation. To aggregate the knowledge into the global model, we average the gradient updates based on their coordinate-wise contributions, as follows:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \frac{1}{\sum_{i \in \mathcal{M}} \boldsymbol{m}_{i,t}} \sum_{i \in \mathcal{M}} \boldsymbol{U}_{i,t} \quad (5)$$

where $\boldsymbol{U}_{i,t}$ is the sparse gradient update from the i-th client's local subspace, and $\sum_{i \in \mathcal{M}} \boldsymbol{m}_{i,t}$ in the divisor accounts for the coordinate-wise contribution. The contribution-based averaging is shown effective in (Vahidian et al., 2021).

Consensus fusion (CF). Given that those malicious parameters serve to recognize backdoor triggers will be deemed unimportant for benign clients, they should not contained in the subspace of benign clients, which accounts for the majority. This observation inspires us to eliminate the malicious parameters using consensus fusion after T rounds of global model training. Formally, the consensus fusion operator returns a vector that satisfies:

$$[\mathcal{T}_\theta(\boldsymbol{m}_{1,T}^{(l)}, \ldots, \boldsymbol{m}_{M,T}^{(l)})]_j = \begin{cases} 1 & \sum_{i=1}^{M} [\boldsymbol{m}_{i,T}^{(l)}]_j \geq \theta \\ 0 & \text{Otherwise} \end{cases} \quad (6)$$

where θ is the threshold for CF, and $[\cdot]_j$ indexes the j-th coordinate of a vector. By applying the global mask produced by CF into the global model, i.e., $\boldsymbol{w}_T \odot \mathcal{T}_\theta(\boldsymbol{m}_{1,T}^{(l)}, \ldots, \boldsymbol{m}_{M,T}^{(l)})$, those parameters that have appearances smaller than θ among all the subspaces are sparsified to 0. In this way, those poisoned parameters will mostly be eliminated, thereby reaching the goal of perturbing the backdoor function. See Figure 5 for an example.

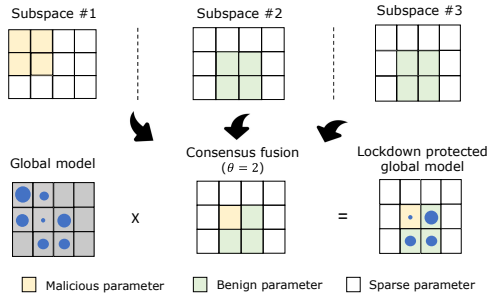

Figure 5. Illustration of consensus fusion (CF). As shown, after multiplying the mask produced by CF, the malicious parameters within the global model, which have fewer appearance in the three subspaces, are mostly sparsified. Therefore, the function for classifying backdoor trigger is perturbed.

5. Experiment

In this section, we present empirical evidence to evaluate the efficacy of Lockdown.

5.1. Experiment Setup

Datasets and models. We experiment on FashionMnist, CIFAR10/CIFAR100 and TinyImagenet datasets. For CIFAR10/CIFAR100, we use ResNet9 model as backbone model. For TinyImagenet, we use a modified ResNet9 via adding a pooling layer after the first convolutional layer to keep the same hidden size before output. For FashionMnist, we use another CNN architecture LeNet5 instead.

Attack methods. We classify the backdoor attack methods in FL into three genres, i.e., *data-level backdoor*, *algorithm-level backdoor* and *advanced backdoor*. Data-level backdoor only allows the malicious clients to modify the raw data, while have no control of the algorithm. On contrary, algorithm-level backdoor allows clients to modify the training algorithm, but not the raw data. The last genre requires extra server-side information (see Appendix B). Indeed,

Table 2. Defense efficacy with varying poison ratio p under CIFAR10.

Methods	Benign Acc (%) ↑					ASR (%) ↓					Backdoor Acc (%) ↑				
(IID)	clean	p=.05	p=.2	p=.5	p=.8	clean	p=.05	p=.2	p=.5	p=.8	clean	p=.05	p=.2	p=.5	p=.8
FedAvg	91.0	91.4	91.1	91.0	90.8	1.6	12.4	19.9	66.1	94.8	88.5	79.6	73.4	32.9	5.1
RLR	86.8	86.7	86.6	86.3	85.5	2.3	2.4	2.4	4.3	25.1	84.6	84.3	83.4	81.7	65.2
Krum	76.3	78.0	75.6	76.4	75.8	4.7	3.9	4.3	4.3	4.9	73.8	74.9	72.9	73.9	73.2
RFA	90.9	91.2	91.1	90.8	90.7	1.6	15.8	20.7	83.7	99.3	88.8	76.8	72.4	15.9	0.7
Trimmed mean	91.0	90.6	91.1	90.9	90.8	1.7	5.0	20.7	61.7	96.2	88.5	84.7	72.0	36.6	3.6
Lockdown	91.2	91.5	91.3	90.7	91.0	1.7	6.1	1.9	1.4	1.5	88.4	84.5	87.5	87.8	87.3

Methods	Benign Acc (%) ↑					ASR (%) ↓					Backdoor Acc (%) ↑				
(Non-IID)	clean	p=.05	p=.2	p=.5	p=.8	clean	p=.05	p=.2	p=.5	p=.8	clean	p=.05	p=.2	p=.5	p=.8
FedAvg	89.0	89.2	89.3	88.8	88.7	1.7	17.3	54.4	86.4	96.7	85.9	74.0	42.5	13.0	3.2
RLR	74.4	74.4	73.6	72.9	72.5	5.8	15.0	40.2	29.5	82.5	69.0	63.1	46.2	51.4	15.3
Krum	42.7	37.4	45.2	43.4	45.1	10.0	5.2	10.4	11.1	10.6	38.6	33.8	40.7	39.3	40.5
RFA	88.8	88.8	88.8	88.3	88.3	2.0	21.4	52.8	90.8	98.7	85.7	70.6	44.3	8.9	1.2
Trimmed mean	88.5	88.4	88.2	88.3	88.3	1.9	25.2	48.4	84.6	96.0	85.4	67.5	47.7	14.7	3.9
Lockdown	85.7	85.9	86.1	84.9	83.7	1.8	14.3	27.5	2.2	3.7	82.7	71.3	60.1	78.4	78.8

the three genres of backdoor attack methods correspond to the weak, medium, strong threat models in Table 1 respectively. Among the data-level backdoor, we simulate three types of attacks methods, i.e., BadNet (Gu et al., 2017), DBA (Xie et al., 2019), and Sinusoidal (Barni et al., 2019). For algorithm-level backdoor, we simulate three types of attack, i.e., Scaling (Bagdasaryan et al., 2020), FixMask, and Snooper. For advanced backdoor, we test Neurontoxin (Zhang et al., 2022) and an adaptive attack Omniscience.

Defense Baselines. We use vanilla FedAvg (McMahan et al., 2016) (without defense) as baseline, and compare Lockdown with four SOTA defenses RLR (Ozdayi et al., 2021), Krum (Blanchard et al., 2017), RFA (Pillutla et al., 2022) and Trimmed mean (Yin et al., 2018).

Evaluation metrics. We design three metrics to evaluate the defense performance of different algorithms, as follows:

- **Benign Acc.** Benign accuracy measures the Top-1 accuracy performance of a model given the benign data inputs (without the presence of a trigger).
- **ASR.** Attack Success Ratio (ASR) measures the ratio of backdoor samples (with trigger) to be classified to the target label. The smaller this metric, the more resilient towards backdoor attack the model is.
- **Backdoor Acc.** Backdoor accuracy measures the Top-1 accuracy of the model given the backdoor sample inputs (their labels during testing are the original one before adding backdoor trigger). We add this metric to evaluate the overall performance, since high backdoor acc means: i) the classification of benign features is well-performing. ii) the perturbation of backdoor trigger is limited.

Simulation setting. We simulate $M = 40$ clients, and data is either evenly distributed to each client (IID setting) or is distributed with Dirichlet distribution (Non-IID setting) following (Hsu et al., 2019). The parameter for Dirichlet distribution is set to 0.5 for the Non-IID partition. To simulate the backdoor attack launched by the malicious clients, we follow (Ozdayi et al., 2021) to randomly choose N clients as attackers whose p (percentage) of data in their local datasets are poisoned. The default backdoor settings for our main experiment are $p = 50\%$ and $N = 4$. We summarize the default simulation setting in Table 3.

Table 3. Default Simulation Setting.

Notation	Meaning	Default Value
M	Total number of clients	40
p	Poison ratio	0.5
N	Attacker number	4
–	Parameter of Dirichlet	0.5

Hyper-parameters. For Lockdown, we fix the overall sparsity to $s = 0.5$, the mask agreement threshold to $\theta = 25$, the shrinkage parameter to $\lambda = 10^{-4}$, and the initial pruning rate to $\alpha_0 = 1$. The robust learning rate threshold for RLR is set to 8. The number of local epochs and batch size are fixed to 2 and 64, respectively. The learning rate and weight decay used in the local optimizer are fixed to 0.1 and 10^{-4}. The total number of communication rounds is fixed to 200. We summarize the default hyper-parameters in Table 4.

Table 4. Default hyper-parameter for Lockdown.

Notation	Meaning	Default Value
α_0	Initial pruning rate	1
θ	Agreement threshold	25
λ	Shrinkage intensity	10^{-4}
s	Overall sparsity	0.5

5.2. Main Evaluation

In this subsection, we show the evaluation results. We use CIFAR10 as the default dataset and BadNet as the default attack method for the main evaluation. The other settings take the default values in Table 3 unless otherwise specified.

Defense efficacy on varying poison ratio. As shown in Table 2, the Attack Success Ratio (ASR) of Lockdown under different data poison ratios are significant lower compared with vanilla FedAvg without defense (up to 93% reduction),

and is also significantly lower than the SOTA backdoor defense solutions (i.e., RLR and Krum, up to 14.2% and 86.1% reduction, respectively). Especially, we observe that Lockdown performs better in reducing ASR when the data poison ratio is high (in Non-IID setting, ASR is 27.5% when $p = 0.2$ while ASR is only 3.7 % when $p = 0.8$). This phenomenon is because the subspaces found by the malicious clients will deviate more from benign clients when a larger amount of backdoor samples are injected in their datasets. Therefore, the malicious subspace will overlap less with benign subspaces, resulting in a better isolation, and also benefit the consensus fusion process. In addition, Lockdown significantly advances backdoor accuracy by up-to 82.2% compared to FedAvg without defense, which implies that the clean model can still recognize the true label of backdoor samples even under attack.

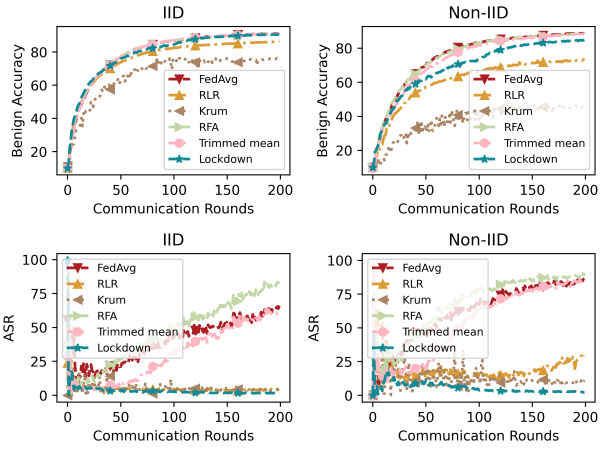

Figure 6. Convergence and defense performance w.r.t communication rounds on CIFAR10 under default setting.

Convergence. The convergence result w.r.t communication rounds is plotted in Figure 6. Compared with IID, Lockdown suffers a slight drop in benign accuracy (by approximately 5%) when the data distribution is Non-IID. This phenomenon can be explained by the heterogeneity of benign subspaces due to the presence of data heterogeneity of benign data. The larger heterogeneity causes more benign parameters to be dropped in the consensus fusion process, resulting in a drop in benign accuracy.

Defense efficacy on varying attacker ratio. We fix the poison ratio to 0.5 and vary the attackers ratio to $\{0, 0.1, 0.2, 0.3, 0.4\}$. The results are shown in Figure 7. As shown, Lockdown *consistently* achieves low ASR (at minimum 50% ASR reduction compared to FedAvg when attackers ratio is 0.4), and high benign accuracy in all groups of experiments (at maximum 5% drop of benign accuracy compared to FedAvg). In contrast, RLR and Krum are *sensitive* to attacker ratio and *fail* in defensing when it is high (no ASR reduction for Krum, and at maximum 10% ASR reduction for RLR when attacker ratio is 0.4).

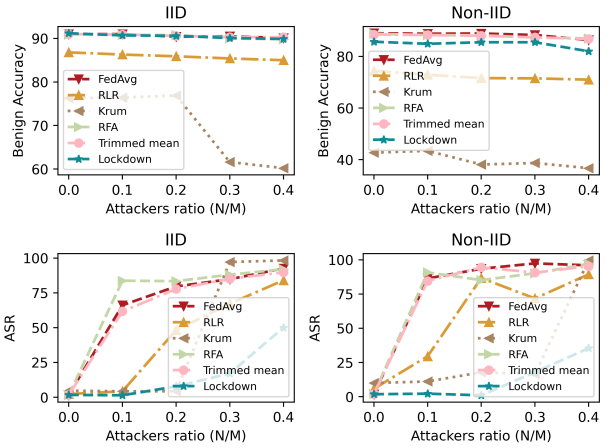

Figure 7. Benign acc/ASR under different attackers ratio.

Defense against data-level backdoor. We simulate attacks with the BadNet, DBA, and sinusoidal method. Our results in Table 5 show that *Lockdown has good generalization ability to data-level backdoor attack.* Lockdown is comparably vulnerable to DBA attack, which increases ASR by 10.7% and 1.8% compared to BadNet attack in the IID and Non-IID setting, while sinusoidal maintain the same level of attack efficiency with BadNet. However, Lockdown still maintains superior defense efficacy ($< 15\%$ ASR) for data-level backdoor attack.

Table 5. Lockdown's defense efficacy on data-level backdoor.

Methods (IID)	Benign Acc(%) ↑	ASR(%) ↓	Backdoor Acc(%) ↑
BadNet	90.7	1.4	87.8
DBA	90.9	3.2	86.3
Sinusoidal	90.9	1.5	87.5
Methods (Non-IID)	**Benign Acc(%) ↑**	**ASR (%) ↓**	**Backdoor Acc(%) ↑**
BadNet	84.9	2.2	78.4
DBA	84.4	12.9	68.8
Sinusoidal	85.0	3.1	80.1

Defense against algorithm-level backdoor. Table 6 shows Lockdown's performance on different algorithm-level backdoor. As can be found, the adaptive attack FixMask, i.e., the malicious clients keep the subspace mask frozen to be the shared initial mask, may not be effective to circumvent Lockdown's defense (with only 1.5% and 2.1% ASR for IID and Non-IID). This is because other benign clients will evolve their subspace per their local data, such that the initial subspace may still have less overlap with the benign client's subspace, and therefore the poisoned weights in initial subspace can still be pruned out by CF operation.

Defense against Omniscience backdoor. As shown in Figure 8, Lockdown is able to defend Neurotoxin attack but is vulnerable to Omniscience attack, the adaptive attack we design. Omniscience is able to infer the global subspace that is produced by CF, and hide its malicious parameters within the global subspace such that they cannot be detected by the server and pruned subsequently. However, we argue that

Table 6. Lockdown's defense on algorithm-level backdoor.

Methods (IID)	Benign Acc(%) ↑	ASR(%) ↓	Backdoor Acc(%) ↑
Vanilla (BadNet)	90.7	1.4	87.8
Scaling (5)	90.2	3.4	85.6
Scaling (10)	89.8	5.7	84.1
FixMask	91.4	1.5	88.8
Snooper	91.6	1.8	88.6
Methods (Non-IID)	**Benign Acc(%) ↑**	**ASR (%) ↓**	**Backdoor Acc(%) ↑**
Vanilla (BadNet)	84.9	2.2	78.4
Scaling (5)	79.2	1.0	73.0
Scaling (10)	74.7	1.3	70.3
FixMask	87.2	2.1	82.7
Snooper	88.3	1.6	84.5

the condition of conducting Omniscience attack is stringent, as it has to acquire either global subspace, or other benign clients' local subspace, which means that it falls into the *Strong* threat model in Table 1.

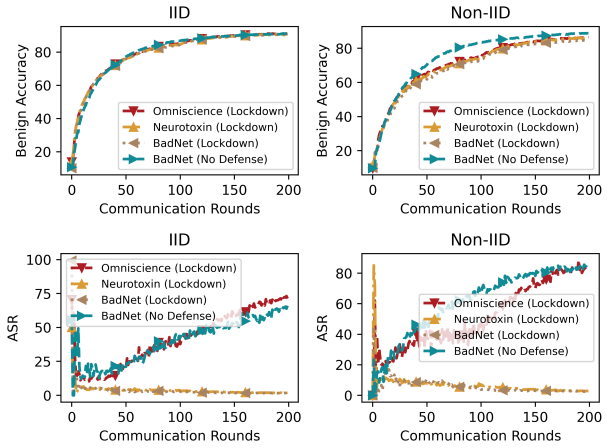

Figure 8. Convergence and defense performance of Lockdown under advanced attack.

Communication and model complexity. As shown in Table 7, we show that Lockdown achieves *smaller communication overhead* (0.5x compared to vanilla FedAvg), since only a small subspace of the entire model gradient needs to be transmitted between server and clients. In addition, the *number of parameters* of the inference model is also *lowered* to 0.49x because the consensus fusion operation prune out the malicious/dummy parameters. Finally, we observe that the *benign accuracy only drops by 0.004x* compared to FedAvg, which indicates that the pruning process will not severely degrade the normal function of the model.

Table 7. Communication and # of parameters for Lockdown under IID CIFAR10 (ResNet9) setting. The communication overhead is the sum of that of $M = 40$ clients in each round.

Methods	Comm Overhead ↓	# of params ↓	Benign Acc(%) ↑
FedAvg	2.10GB (1x)	6.57M (1x)	91.0 (1x)
Lockdown	1.05 GB (0.5x)	3.25M (0.49x)	90.7 (0.996x)

Adaptive timing attack. We show the evaluation results under adaptive timing attack. In our simulation, we assume the malicious clients only perform attack in the initial 50 rounds. Our results shows that Omniscience can recover from poisoning if attack is only conducted in early rounds. The benign gradient update from clients concentrating on the benign subspace can automatically recover the poisoned parameters located in this subspace. The same recovery phenomenon is also reported in (Zhang et al., 2022). However, Neurotoxin (Zhang et al., 2022), which aims to prolong the timespan of backdoor effect, no longer works under Lockdown defense. This is because the poisoned parameters Neurotoxin inject outside of the benign subspace will be automatically pruned by Lockdown.

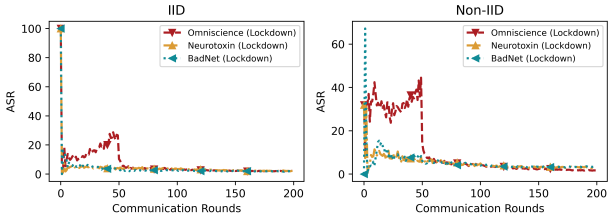

Figure 9. Defense performance under adaptive timing.

Performance on varying dataset. We show our evaluation results on FashionMnist, CIFAR10/100 and TinyImagenet in Table 8. As shown, Lockdown achieves SOTA defense efficacy (compared with FedAvg without defense, with up-to 96,1%, 84.2%, 73.8%, and 87.4% reduction of ASR respectively on the four datasets), and maintains a reasonable loss of benign accuracy compared to FedAvg without defense (with up-to 1%, 3.9%, 3.9% and 3.4% drop).

Table 8. Performance on varying datasets.

Dataset	Methods	Benign Acc ↑	ASR ↓	Backdoor Acc ↑
FashionMnist (IID)	FedAvg	**91.2**	96.9	2.9
	RLR	90.4	92.0	7.3
	Krum	85.0	**0.0**	62.7
	Lockdown	90.6	0.8	**79.1**
FashionMnist (Non-IID)	FedAvg	**90.3**	96.3	3.5
	RLR	87.4	94.1	5.1
	Krum	74.3	1.8	71.5
	Lockdown	89.3	**1.2**	**81.7**
CIFAR10 (IID)	FedAvg	**91.0**	66.1	32.9
	RLR	86.3	4.3	81.7
	Krum	76.4	4.3	73.9
	Lockdown	90.7	**1.4**	**87.8**
CIFAR10 (Non-IID)	FedAvg	**88.8**	86.4	13.0
	RLR	72.9	29.5	51.4
	Krum	43.4	11.1	39.3
	Lockdown	84.9	**2.2**	**78.4**
CIFAR100 (IID)	FedAvg	**70.0**	75.7	21.1
	RLR	61.0	**1.9**	55.5
	Krum	26.9	98.8	0.7
	Lockdown	66.9	**1.9**	**63.3**
CIFAR100 (Non-IID)	FedAvg	**67.7**	74.8	21.0
	RLR	53.9	36.0	37.6
	Krum	20.6	**1.3**	19.8
	Lockdown	63.8	2.5	**59.5**
TinyImagenet (IID)	FedAvg	**18.5**	97.2	0.9
	RLR	8.2	67.4	3.1
	Krum	3.4	99.9	0.0
	Lockdown	15.9	**55.2**	**9.0**
TinyImagenet (Non-IID)	FedAvg	**17.6**	96.9	1.0
	RLR	7.4	81.3	2.0
	Krum	2.7	99.9	0.0
	Lockdown	14.2	**9.5**	**12.8**

5.3. Ablation and Hyper-parameter Sensitivity Analysis

Ablation study is moved to Appendix F. For sensitivity analysis, we tune three key hyper-parameters of Lockdown to demonstrate their impacts. Our key findings are: i) setting a proper sparsity s can increase the model's robustness. ii) setting a proper shrinkage intensity λ is necessary for effective subspace searching, but too large intensity will hurt the model's normal function. and iii) agreement threshold θ should be set sufficiently large to filter out the malicious parameters. Details are moved to Appendix G.

6. Conclusion

In this paper, we propose Lockdown, a backdoor defense solution for federated learning based on the idea of isolated subspace training. Empirical evidence shows that Lockdown can significantly reduce the risk of malicious backdoor attacks without sacrificing much on benign accuracy. Future works include studying how to generalize Lockdown to other federated learning settings, e.g., decentralized FL (Hu et al., 2019; Dai et al., 2022), personalized FL (Fallah et al., 2020; Deng et al., 2020; T Dinh et al., 2020), etc.

References

Acar, D. A. E., Zhao, Y., Navarro, R. M., Mattina, M., Whatmough, P. N., and Saligrama, V. Federated learning based on dynamic regularization. *arXiv preprint arXiv:2111.04263*, 2021.

Alfarra, M., Pérez, J. C., Shulgin, E., Richtárik, P., and Ghanem, B. Certified robustness in federated learning. *arXiv preprint arXiv:2206.02535*, 2022.

Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., and Shmatikov, V. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 2938–2948. PMLR, 2020.

Barni, M., Kallas, K., and Tondi, B. A new backdoor attack in cnns by training set corruption without label poisoning. In *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 101–105. IEEE, 2019.

Bhagoji, A. N., Chakraborty, S., Mittal, P., and Calo, S. Analyzing federated learning through an adversarial lens. In *International Conference on Machine Learning*, pp. 634–643. PMLR, 2019.

Bibikar, S., Vikalo, H., Wang, Z., and Chen, X. Federated dynamic sparse training: Computing less, communicating less, yet learning better. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 6080–6088, 2022.

Blanchard, P., El Mhamdi, E. M., Guerraoui, R., and Stainer, J. Machine learning with adversaries: Byzantine tolerant gradient descent. *Advances in Neural Information Processing Systems*, 30, 2017.

Cao, X., Fang, M., Liu, J., and Gong, N. Z. Fltrust: Byzantine-robust federated learning via trust bootstrapping. *arXiv preprint arXiv:2012.13995*, 2020.

Cao, X., Zhang, Z., Jia, J., and Gong, N. Z. Flcert: Provably secure federated learning against poisoning attacks. *IEEE Transactions on Information Forensics and Security*, 17: 3691–3705, 2022.

Chow, K.-H. and Liu, L. Perception poisoning attacks in federated learning. In *2021 Third IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, pp. 146–155. IEEE, 2021.

Cohen, J., Rosenfeld, E., and Kolter, Z. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, pp. 1310–1320. PMLR, 2019.

Dai, R., Shen, L., He, F., Tian, X., and Tao, D. Dispfl: Towards communication-efficient personalized federated learning via decentralized sparse training. *arXiv preprint arXiv:2206.00187*, 2022.

Deng, Y., Kamani, M. M., and Mahdavi, M. Adaptive personalized federated learning. *arXiv preprint arXiv:2003.13461*, 2020.

Evci, U., Gale, T., Menick, J., Castro, P. S., and Elsen, E. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, pp. 2943–2952. PMLR, 2020.

Fallah, A., Mokhtari, A., and Ozdaglar, A. Personalized federated learning: A meta-learning approach, 2020.

Ghosh, A., Hong, J., Yin, D., and Ramchandran, K. Robust federated learning in a heterogeneous environment. *arXiv preprint arXiv:1906.06629*, 2019.

Gu, T., Dolan-Gavitt, B., and Garg, S. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

Hsu, T.-M. H., Qi, H., and Brown, M. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

Hu, C., Jiang, J., and Wang, Z. Decentralized federated learning: A segmented gossip approach. *arXiv preprint arXiv:1908.07782*, 2019.

Huang, T., Liu, S., Shen, L., He, F., Lin, W., and Tao, D. Achieving personalized federated learning with sparse local models. *arXiv preprint arXiv:2201.11380*, 2022.

Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S., Stich, S., and Suresh, A. T. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pp. 5132–5143. PMLR, 2020.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 2018.

Liu, S., Chen, T., Chen, X., Atashgahi, Z., Yin, L., Kou, H., Shen, L., Pechenizkiy, M., Wang, Z., and Mocanu, D. C. Sparse training via boosting pruning plasticity with neuroregeneration. *arXiv preprint arXiv:2106.10404*, 2021a.

Liu, S., Yin, L., Mocanu, D. C., and Pechenizkiy, M. Do we actually need dense over-parameterization? in-time over-parameterization in sparse training. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 6989–7000. PMLR, 2021b.

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., et al. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2016.

Mo, F., Haddadi, H., Katevas, K., Marin, E., Perino, D., and Kourtellis, N. Ppfl: privacy-preserving federated learning with trusted execution environments. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*, pp. 94–108, 2021.

Mocanu, D. C., Mocanu, E., Stone, P., Nguyen, P. H., Gibescu, M., and Liotta, A. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature Communications*, 9 (1):2383, 2018.

Ozdayi, M. S., Kantarcioglu, M., and Gel, Y. R. Defending against backdoors in federated learning with robust learning rate. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 9268–9276, 2021.

Pillutla, K., Kakade, S. M., and Harchaoui, Z. Robust aggregation for federated learning. *IEEE Transactions on Signal Processing*, 70:1142–1154, 2022.

Shah, D., Dube, P., Chakraborty, S., and Verma, A. Adversarial training in communication constrained federated learning. *arXiv preprint arXiv:2103.01319*, 2021.

Sifaou, H. and Li, G. Y. Robust federated learning via over-the-air computation. In *2022 IEEE 32nd International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6. IEEE, 2022.

Smilkov, D., Thorat, N., Kim, B., Viégas, F., and Wattenberg, M. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

T Dinh, C., Tran, N., and Nguyen, J. Personalized federated learning with moreau envelopes. *Advances in Neural Information Processing Systems*, 33:21394–21405, 2020.

Tahmasebian, F., Lou, J., and Xiong, L. Robustfed: a truth inference approach for robust federated learning. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pp. 1868–1877, 2022.

Tolpegin, V., Truex, S., Gursoy, M. E., and Liu, L. Data poisoning attacks against federated learning systems. In *European Symposium on Research in Computer Security*, pp. 480–501. Springer, 2020.

Vahidian, S., Morafah, M., and Lin, B. Personalized federated learning by structured and unstructured pruning under data heterogeneity. *arXiv preprint arXiv:2105.00562*, 2021.

Wang, H., Sreenivasan, K., Rajput, S., Vishwakarma, H., Agarwal, S., Sohn, J.-y., Lee, K., and Papailiopoulos, D. Attack of the tails: Yes, you really can backdoor federated learning. *arXiv preprint arXiv:2007.05084*, 2020.

Xie, C., Huang, K., Chen, P.-Y., and Li, B. Dba: Distributed backdoor attacks against federated learning. In *International Conference on Learning Representations*, 2019.

Xie, C., Chen, M., Chen, P.-Y., and Li, B. Crfl: Certifiably robust federated learning against backdoor attacks. In *International Conference on Machine Learning*, pp. 11372–11382. PMLR, 2021.

Yin, D., Chen, Y., Kannan, R., and Bartlett, P. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning*, pp. 5650–5659. PMLR, 2018.

You, H., Li, C., Xu, P., Fu, Y., Wang, Y., Chen, X., Baraniuk, R. G., Wang, Z., and Lin, Y. Drawing early-bird tickets: Towards more efficient training of deep networks. *arXiv preprint arXiv:1909.11957*, 2019.

Zhang, X., Hong, M., Dhople, S., Yin, W., and Liu, Y. Fedpd: A federated learning framework with optimal rates and adaptivity to non-iid data. *arXiv preprint arXiv:2005.11418*, 2020.

Zhang, Z., Panda, A., Song, L., Yang, Y., Mahoney, M., Mittal, P., Kannan, R., and Gonzalez, J. Neurotoxin: Durable backdoors in federated learning. In *International Conference on Machine Learning*, pp. 26429–26446. PMLR, 2022.

Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., and Chandra, V. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

Zizzo, G., Rawat, A., Sinn, M., and Buesser, B. Fat: Federated adversarial training. *arXiv preprint arXiv:2012.01791*, 2020.

A. Implementation Details.

Decay pruning rate with cosine annealing. In SubspaceSearch() function, we let the clients prune out $\alpha_t\%$ of parameters and recover the same amount of parameters to search for the subspace that fits their data. The parameter $\alpha_t\%$ is decayed with the initial rate α_0 with cosine annealing, which can be formalized as follows:

$$\alpha_t\% = 0.5 \times \alpha_0 \times \left(1 + \cos\left(\frac{\mathrm{t}}{T_{end}}\pi\right)\right) \tag{7}$$

where t is the number of communication round, and T_{end} is the round that the mask searching is ended (notice that $\alpha_{T_{end}}\% = 0$). In our implementation, we set $T_{end} = T/2$ since we empirically observe that the subspace has already been fixed after early rounds of training, which may be explained by the early bird ticket hypothesis in (You et al., 2019).

ERK sparsity initialization. We use Erdős–Rényi Kernel (ERK) (Evci et al., 2020), an empirical sparsity distribution technique, to distribute sparsity to different layers of a model. Specifically, the active parameters of the convolutional layer initialized by ERK are proportional to $1\frac{n^{1-1}+n^l+w^l+h^l}{n^{l-1}*n^l*w^l*h^l}$, where n^{l-1}, n_l w^l and h^l respectively specify the number of input channels, output channels and kernel's width and height in the l-th layer. For the linear layer, the number of active parameters scale with $1\frac{\mathrm{n}^{l-1}+\mathrm{n}^l}{\mathrm{n}^{l-1}*\mathrm{n}^l}$ where n^{l-1} and n^l are the number of neurons in the $l-1$-th and l-th layer. ERK initialization, in essence, gives more sparsity to those layers with a larger number of parameters while pruning less on those small layers.

Subspace pruning. In the mask searching process, we use parameter's magnitude to guide the pruning of model parameters. We present the PyTorch style code in Algorithm 3 to illustrate the pruning process and, correspondingly, the update of mask. Note that we only prune out parameters that are within the current subspace. Therefore, in line 6, we set the parameters that are out of the subspace to a very large value to prevent selecting them. After that, we filter out those parameters with the smallest $\alpha_t\%$ magnitude and prune them out of the subspace.

Algorithm 3 PyTorch style code for pruning and recovery

1: **function Prune_subspace**($\alpha_t\%$, $\boldsymbol{w}_{i,t,K}$, $\boldsymbol{m}_{i,t}$)
2: Init layer sparsity $\{s_l\}$ given overall sparsity s with ERK
3: $\boldsymbol{m}_{i,t+\frac{1}{2}} = \boldsymbol{m}_{i,t}$
4: **for** $l = 0, 1, \ldots, L-1$ **do**
5: $num_{prune} = \alpha_t\% \times$ # of params in the l-th layer
6: $sort\_value$ = torch.where($\boldsymbol{m}^{(l)}_{i,t} == 1$, torch.abs($\boldsymbol{w}^{(l)}_{i,t,K}$), 100000×torch.ones_like($\boldsymbol{w}^{(l)}_{i,t,K}$))
7: $\_, idx$ = torch.sort(($sort\_value$).view(-1))
8: $\boldsymbol{m}^{(l)}_{i,t+\frac{1}{2}}$.view($-1$)[$idx$[: num_{prune}]] $= 0$
9: **end for**
10: Return $\boldsymbol{m}_{i,t+\frac{1}{2}}$
11: **end function**
12:
13: **function Recover_subspace**($\alpha_t\%$, $\boldsymbol{w}_{i,t,K}$, $\boldsymbol{m}_{i,t+\frac{1}{2}}$)
14: Derive gradient $\nabla f_i(\boldsymbol{w}_{i,t,K} \odot \boldsymbol{m}_{i,t+\frac{1}{2}})$ with one pass of local data
15: $\boldsymbol{m}_{i,t+1} = \boldsymbol{m}_{i,t+\frac{1}{2}}$
16: **for** $l = 0, 1, \ldots, L-1$ **do**
17: $num_{prune} = \alpha_t\% \times$ # of params in the l-th layer
18: $sort\_value$ = torch.where($\boldsymbol{m}^{(l)}_{i,t+\frac{1}{2}} == 0$, torch.abs($\nabla f^{(l)}_i(\boldsymbol{w}_{i,t,K} \odot \boldsymbol{m}_{i,t+\frac{1}{2}})$), -100000×torch.ones_like($\boldsymbol{w}^{(l)}_{i,t,K}$))
19: $\_, idx$ = torch.sort(($sort\_value$).view(-1), descending=True)
20: $\boldsymbol{m}^{(l)}_{i,t+1}$.view($-1$)[$idx$[: num_{prune}]] $= 1$
21: **end for**
22: Return $\boldsymbol{m}_{i,t+1}$
23: **end function**

Subspace recovery. After pruning, we recover the same amount of parameters to explore other parameters outside the subspace. Following (Evci et al., 2020), we use gradient information of the pruned model to guide the recovery process. Here we only recover the parameters out of the current subspace, and therefore we set the $sort\_value$ of the parameters within the current subspace to a sufficiently small value, as shown in line 18 in Algorithm 3. Subsequently, we sort in descending to obtain the parameters with the largest-$\alpha_t\%$ gradient magnitude, and recover them by updating masks.

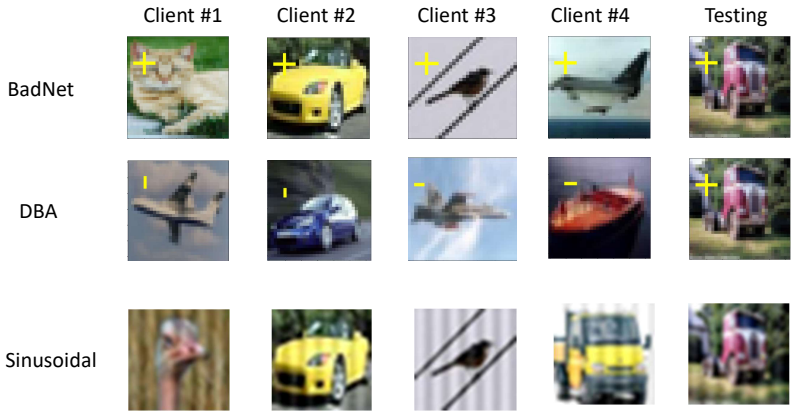

Figure 10. Examples of BadNet, DBA, and Sinusoidal attack. Labels of poison samples are manipulated to the target label (e.g., a horse).

B. Attack Methods

Table 9. Application of attack methods in threat models. "✓" corresponds to be applicable while "✗" corresponds to be not applicable.

Attack methods /	Threat models		
	weak	medium	strong
BadNet	✓	✓	✓
DBA	✓	✓	✓
Sinusoidal	✓	✓	✓
Scaling	✗	✓	✓
FixMask (adaptive)	✗	✓	✓
Snooper (adaptive)	✗	✓	✓
Neurotoxin	✗	✗	✓
Omniscience (adaptive)	✗	✗	✓

As we mention in the main body, we classify the attack model into data-level attack and algorithm-level backdoor. We in the following give brief description of each data-level backdoor that we simulate in federated learning setting.

- **BadNet**. BadNet is the earliest, and also the simplest backdoor attack first proposed in (Gu et al., 2017). To perform BadNet attack, the malicious client simply add the same backdoor trigger on some of the data samples, and modify the label of these poisoned samples to the target label. In test time, the malicious clients can place the backdoor trigger on the test samples, such that the victim model can produce the target output no matter what the original test samples are.

- **DBA.** DBA (Xie et al., 2019) is a backdoor attack specifically targeted on FL. To perform DBA attack, the authors decompose the backdoor trigger into several local pattern, and assign the local pattern to different clients to poison their local data. For test time, the attacker will interpose the completed trigger on top of the test samples they want to manipulate. It is suggested by the authors that DBA is substantially more persistent and stealthy against FL. In our simulation, we decompose the "plus" trigger into 4 local patterns, and let each malicious client to be assigned each local pattern.

- **Sinusoidal attack.** Sinusoidal attack (Barni et al., 2019) shares a similar perspective with BadNet, which also utilize the same trigger for all the malicious clients to poison their samples. However, the backdoor trigger they use is a horizontal sinusoidal signal defined by $v(i, j) = \Delta \sin(2\pi j f/m), 1 \leq j \leq m, 1 \leq i \leq l$, for a certain frequency f. The authors claim that this design of trigger i) is weal enough to ensure the stealthiness of the attack, but also ii) be detectable in the same (or similar) feature space used by the network to classify the pristine samples. In our simulation, we adopt the default hyper-parameter $\delta = 20$ and $f = 6$ for performing this attack.

Examples of these data-level attacks are visually shown in Figure 10.

In the following we give brief description on the algorithm-level backdoor that has been simulated in this paper.

- **Scaling**. The basic idea of Scaling (Bagdasaryan et al., 2020) is to enlarge the gradient update when a malicious client return its update to server. This mechanism allows the malicious client to enlarge its gradient's impact on the global model, and therefore is effective when the poison ratio and attacker number are small.

- **FixMask**. FixMask is an adaptive attack method specifically targeting Lockdown. In Lockdown, the malicious clients are assumed to faithfully search for their subspace using their local data. For FixMask attack, the malicious clients freeze their mask to be the initial mask that is shared by all the clients in round $t = 0$, and refuse to change afterwards.

- **Snooper**. Snooper is another adaptive attack method for Lockdown. For snooper attack, the malicious clients seek to find the consensus subspace using its limited gradient information, and project its malicious update to the found subspace. Particularly, snooper uses the following heuristic to search for the consensus mask: i) it maintains the topk coordinates of the absolute subspace gradient that was downloaded from the server, and delete other coordinates from its subspace, ii) it randomly explores other coordinates that are out of its current subspace. This adaptive attack is based on the idea that the benign subspace should have large l2 norm gradient update, and therefore can be detected in an exploration and exploitation manner.

Particularly, we want to emphasize that the data-level and algorithm-level backdoor can potentially be combined together to produce better attack performance. However, since this paper focus on the defense aspect, we leave a more thorough study of the attack model future work. We also include two advanced attack algorithms that can only be conducted given extra server information in addition to permission on manipulation of the attacker's own training process and data.

- **Neurotoxin**. Neurotoxin proposed in (Zhang et al., 2022) explores a durable attack method in the scenario that the attackers can only participate limited rounds. Their main observations in the limited participation case are that i) the benign update can recover the global model after attacker ceases attack. ii) the majority of the l2 norm of the aggregated benign update is contained in a small number of coordinates (Let's call these benign coordinates). Utilizing the above observations, the authors propose Neurotoxin, which is to let the malicious clients project their gradient update to the subspace excluding the global coordinates. By this means, the projected updates from the malicious clients are mostly embedded to the coordinates that have less perturbation by the benign updates (which focus on the benign coordinates) after ceasing attack. However, Neurotoxin cannot escape Lockdown defense in principle. There are mainly two reasons. i) Lockdown only broadcast to the clients some coordinates weights (equivalently, some coordinates of gradient update) as per their subspace (see line 6 in algorithm 1. Therefore, Neurotoxin cannot obtain the top-k coordinate of the server gradient as benign coordinates. ii) Lockdown requires clients to report the subspace that they want to update, and the subspace that are substantially different from others will be pruned afterwards. In other words, if the attackers adopt neurotoxin to choose the subspace that excludes the benign coordinates, their subspace can be easily identified by comparing with other benign client's subspace, and therefore will be pruned out. In our simulation, we assume Neurotoxin can acquire the server gradient update by some means. Therefore, it is classified as an attack method for *strong* threat model. In our simulation, we set its hyper-parameter mask ratio to be 0.5.

- **Omniscience**. This is an adaptive attack that assumes the knowledge of Lockdown and try to break it. The main idea is to assume the client's has knowledge of the consensus subspace after going through consensus fusion, and project their gradient update into this subspace. This efficiently avoids the malicious weights to be pruned out by the consensus fusion operation. However, the requirement of conducting this attack is very stringent. The malicious client needs to have knowledge of the consensus subspace, which either is leaked from server, or is computed if other clients' subspace is known by the attacker. Neither of this condition is easy to establish for an attacker in a federated learning system.

In summary, we show in Table 9 the attack methods we can perform in specific threat models.

C. Defense Methods

In this section, we give a brief description of the defense baseline we compare against.

- **RLR**. RLR proposed in (Ozdayi et al., 2021) utilizes coordinate-wise server learning rate to inverse the gradient coordinates in which different clients have different sign. Their observation is that the malicious coordinates tends to be those coordinates that have conflicting sign in gradient while for the benign coordinates that are not poisoned, most of the clients will agree with their sign. Therefore, by looking at the gradient update from clients, the server is able to identify the malicious coordinates and subsequently inverse its sign in the aggregation phase. However, the malicious clients may be able to launch adaptive attack to this defense if he knows the gradient update downloaded from server.

- **RFA**. Aiming at defense against corrupted updates from clients, RFA (Pillutla et al., 2022) utilizes the concept of geometric medium to aggregate the gradient update from clients. Geometric medium avoids the gradient that has excessively large norm (usually is the malicious one) to impact too much on the averaging process. Specifically, when doing aggregation, instead of directly averaging the uploaded gradient, the server aims to obtain global model v that minimizes: $\sum_{i=1}^{m} \|v - w_i\|$, and w_i is the uploaded local model. This optimization problem is solved by the Smoothed Weiszfeld Algorithm. Some similar techniques is also studied in (Sifaou & Li, 2022), (Ghosh et al., 2019) and (Cao et al., 2020).

- **Krum**. Targeting Byzantine attack, Krum (Blanchard et al., 2017) adopts the idea of finding the gradient update that is closest to its $n - f - 2$ neighbours such that it can ensure (α, f)-Byzantine resilience where α is the angle depends on the ratio of the deviation over the gradient ,f is the number of attackers. Specifically, Krum aims to find the the i^*-th client that minimize $s(i) = \sum_{i \to j} \|V_i - V_j\|^2$ where $i \to j$ denotes the set of i'th client's $n - f - 2$ closest neighbours, and V_i denotes the gradient update from client i. After identifying i^*, Krum returns V_{i^*} as the robust gradient used for aggregation.

- **Trimmed mean**. Trimmend mean is proposed in (Yin et al., 2018) to counter byzantine failures in the distributed machine learning scenario. Their high level idea is to exclude the outlier gradient value when doing aggregation. Specifically, before aggregation, the server coordinate-wise trims the TopK gradient and the bottomK gradient among those uploaded gradient. After trimming, the server assume the outlier has been trimmed, and directly average the clean gradient. In our simulation, we set the trim ratio to be 0.1.

D. Security Analysis

With support of our empirical evidence provided in Section 5, we make the following observations on Lockdown's security performance. Lockdown can successfully defend all the data-level attack, i.e., the attack falls in to the scope of *weak* threat models. For the algorithm-level attacks, we have incorporated two adaptive attacks targeting on Lockdown and a gradient scaling method into study. Our results show that Lockdown can also defend all the attacks we have tested. However, since the algorithm-level attacks are more adaptive, we do not make guarantee that Lockdown is unbreakable by any algorithm-level attacks, especially those that are specially designed for Lockdown. For advanced attack that allows attacker to acquire server's information, we create another adaptive attack Omniscience that can successfully circumvent Lockdown's defense. Performing Omniscience attack needs the attacker to know about the consensus subspace. However, it is challenging, if not impossible, for the attacker to infer the consensus subspace, since only a subset of the server gradient update is distributed to clients, further constraining the global information access of the attackers. We have designed two adaptive algorithm-level attacks named FixMask and Snooper to test the attackers' ability to infer the consensus subspace. However, both these two attacks fail against Lockdown defense.

E. Visualization

In Figure 11, we add additional experiments to visualize the gradient w.r.t the input of the first layer, which visually explains how different semantic information within the input image contributes to activating the target output neuron.

In Figure 12, we visualize the projected parameters produced by Lockdown. The experiment is conducted on MNIST with a two-layer MLP model. After reducing its output dimension and reshaping it into the original input, we plot the projected absolute weights of the first layer of MLP. As found, by projecting the global weights into malicious client's subspace (left), the corresponding connectivity that joint the backdoor trigger still present. However, by projecting the global weights into one of the benign client's subspace (middle), the backdoor trigger no longer connects with large absolute weights. The same phenomenon is observed for the consensus subspace after going through consensus fusion (right).

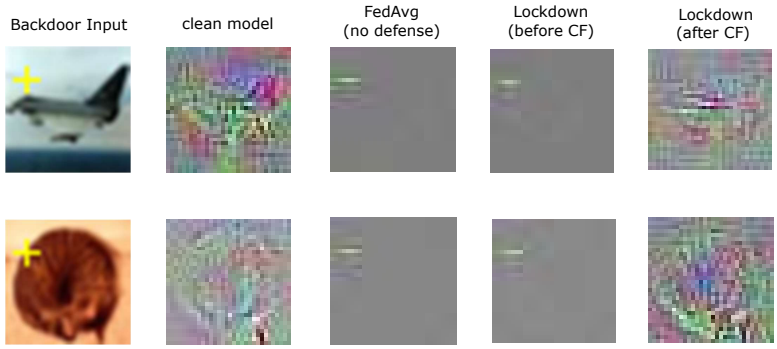

Figure 11. Smooth grad (Smilkov et al., 2017) visualization of models given backdoor input. The first column is data input with backdoor trigger. The incoming column subsequently demonstrate the gradient with respect to the input of i) a model without being poisoned. ii) a model trained by FedAvg with poisoned data, iii) Lockdown's global model under poisoning before going through consensus fusion (CF) and iv) Lockdown's final model. A clean model emphasize the correct semantic within the input, e.g., wing of a plane, while a poisoned model emphasizes the yellow "plus" backdoor trigger.

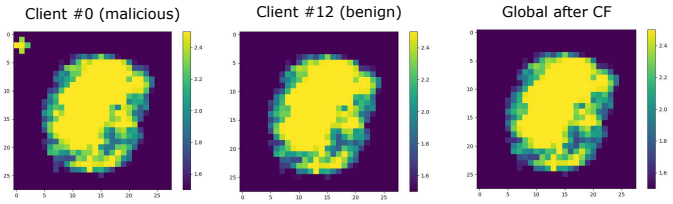

Figure 12. Visualization of absolute global weights after projecting into the local or global subspace. Left: projecting into local subspace of a malicious client. Middle: projecting into local subspace of a benign client. Right: projecting into consensus subspace produced by consensus fusion. The brighter the color is, the feature locates in that part is more important. The bright backdoor trigger "+" is not visible in the middle and right image. See more details in the main text.

F. Ablation study

Then we evaluate the function of each component of Lockdown on CIFAR10. BadNet is the default attack method. Default settings are used unless otherwise specified.

Average w/ mask vs. direct average. Recall in Eq. (5), we apply aggregation based on the mask contribution. For comparison, we replace the aggregation rule with Average, i.e., the vanilla aggregation rule in FedAvg. The comparison result is available in Table 10, which shows that average w/ mask significantly outperforms direct average in terms of reducing ASR (up-to 33.3 % reduction), though it slightly degrades benign accuracy (at maximum 2.9% drop).

Table 10. Ablation study for aggregation operation.

Methods (IID)	Benign Acc(%) ↑	ASR(%) ↓	Backdoor Acc(%) ↑
Average	**91.4**	6.8	84.2
Average w/ mask (ours)	90.7	**1.4**	**87.8**
Methods (Non-IID)	Benign Acc(%) ↑	ASR(%) ↓	Backdoor Acc(%) ↑
Average	**87.8**	35.5	59.6
Average w/ mask (ours)	84.9	**2.2**	**78.4**

Gradient-based recovery vs. random recovery. In SubspaceSearch(·) of Algorithm 2, we use gradient magnitude to guide the recovery of parameters. In Table 11, we show the empirical comparison between the gradient-based recovery and random recovery. The results showcase that recovery with the gradient can significantly reduce the ASR (by up-to 15% reduction) though the benign acc of the model suffered a little bit (by up-to 4% drop). This is because gradient magnitude tends to guide the subspace searching process to acquire heterogeneous subspaces for clients with different training data. With more heterogeneous subspaces, the knowledge transferring between clients will be deterred since their the subspace

overlap is small, which leads to the degradation of benign acc. On the other hand, small subspace overlap can also facilitate the process of de-poisoning by consensus fusion, which leads to a reduction of ASR.

Table 11. Ablation study for parameters recovery implementation.

Methods (IID)	Benign Acc(%) ↑	ASR(%) ↓	Backdoor Acc(%) ↑
Random recovery	**91.1**	13.0	79.7
Recovery w/ gradient (ours)	90.7	**1.4**	**87.8**
Methods (Non-IID)	**Benign Acc(%) ↑**	**ASR(%) ↓**	**Backdoor Acc(%) ↑**
Random recovery	**88.9**	17.2	74.3
Recovery w/ gradient (ours)	84.9	**2.2**	**78.4**

ERK initialization vs. uniform initialization. In the SubspaceInit() function of Algorithm 2, we use ERK to allocate the sparsity of each layer in a model. To justify the necessity of ERK initialization, we replace the ERK initialization with uniform initialization, which uniformly allocates sparsity to each layer. As shown in Table 12, uniform initialization will largely compromise the benign accuracy and slightly increase the ASR. This justifies that the sparsity should be set larger for the layer with a larger number of trainable parameters, while for the layers with a smaller amount of parameters, the sparsity cannot be set too large.

Table 12. Ablation study for sparsity initialization.

Methods (IID)	Benign Acc ↑	ASR ↓	Backdoor Acc ↑
Uniform	70.2	6.3	64.5
ERK (ours)	**90.7**	**1.4**	**87.8**
Methods (Non-IID)	**Benign Acc ↑**	**ASR ↓**	**Backdoor Acc ↑**
Uniform	77.8	4.6	73.5
ERK (ours)	**84.9**	**2.2**	**78.4**

Consensus fusion (CF). In Figure 13, we demonstrate the necessity of conducting consensus fusion under different poison ratios. With consensus fusion, benign accuracy is significantly increased by up-to 60% while the ASR is reduced by up-to 80%. This result demonstrates that masking out some malicious/dummy parameters can perturb the backdoor function and thereby curing the poisoned global model.

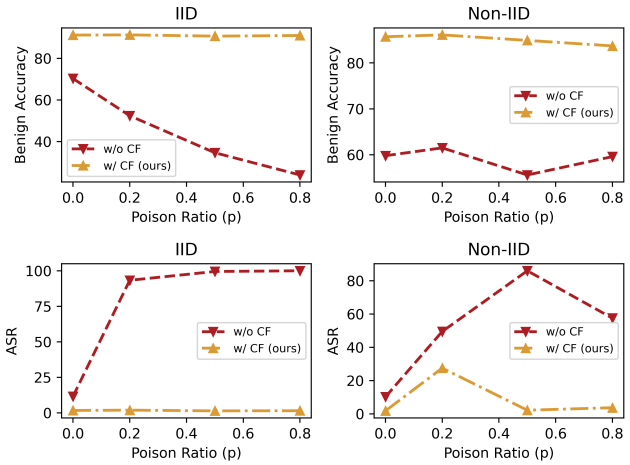

Figure 13. Impact of consensus fusion of Lockdown.

G. Hyper-parameter Sensitivity Analysis

In this section, we perform hyper-parameter sensitivity analysis for lockdown. The evaluation is conducted on CIFAR10 under the default simulation setting in Table 3 unless otherwise specified.

Sparsity s. In Table 13, we set other hyper-parameters as default and tune the sparsity to different levels. As shown, Lockdown loses its defense efficacy when sparsity is low. This phenomenon is easy to understand since Lockdown reduces

to FedAvg when sparsity is 0. In addition, with larger sparsity, the benign accuracy of the model is also suffering due to the reduction of trainable parameters. Therefore, there exists a tradeoff for the sparsity of Lockdown. Larger sparsity brings lower model complexity, smaller communication overhead, and also lower ASR, but at the cost of losing benign accuracy.

Table 13. Performance of Lockdown under different sparsity s.

s (IID)	Benign Acc ↑	ASR ↓	# of params ↓
0.2	**91.3**	46.0	5.29M
0.5	90.7	**1.4**	3.26M
0.8	90.4	3.2	**1.29M**
s (Non-IID)	Benign Acc ↑	ASR ↓	# of params ↓
0.2	**88.9**	74.6	5.29M
0.5	84.9	**2.2**	3.25M
0.8	85.5	2.6	**1.23M**

Shrinkage intensity λ. We fix poison ratio $p = 0.2$ and attacker number $N = 4$, and tune shrinkage intensity λ to see its impact on the defense efficacy. As shown in Table 14, larger shrinkage intensity will compromise benign accuracy, but also guarantee better defense efficacy, i.e., reducing more ASR. This phenomenon justifies that sufficient shrinkage intensity is necessary in order for the subspaces of clients to be found efficiently, therefore enabling effective pruning of malicious parameters subsequently in the consensus fusion phase.

Table 14. Performance under different shrinkage intensity λ.

λ (IID)	Benign Acc ↑	ASR ↓	Backdoor Acc ↑
0	90.8	5.2	83.5
10^{-5}	**91.3**	13.7	78.1
10^{-4}	**91.3**	**2.5**	**87.5**
10^{-3}	86.2	4.0	80.8
λ (Non-IID)	Benign Acc ↑	ASR ↓	Backdoor Acc ↑
0	85.5	60.2	35.8
10^{-5}	**86.5**	53.3	42.1
10^{-4}	86.1	27.5	60.1
10^{-3}	81.3	**9.0**	**73.1**

Consensus fusion threshold θ. In Figure 14, we tune the CF threshold θ to see its impact on different settings of attacker number N. In all settings of N, we see that: i) θ should not be set to be too small; otherwise, the benign accuracy would be lower, and the ASR will be higher. ii) θ also should not be set too large; otherwise, it will severely compromise benign accuracy, but the reduction of ASR will not be too significant. Per our results, the agreement threshold should be chosen carefully according to the number of attackers, which of course, is unknown in most cases. However, given that the attackers within the system should not take up a large portion, θ set to be 50% of the total number of clients will be sufficient to counteract the effect of backdoor attack in a general attack scenario.

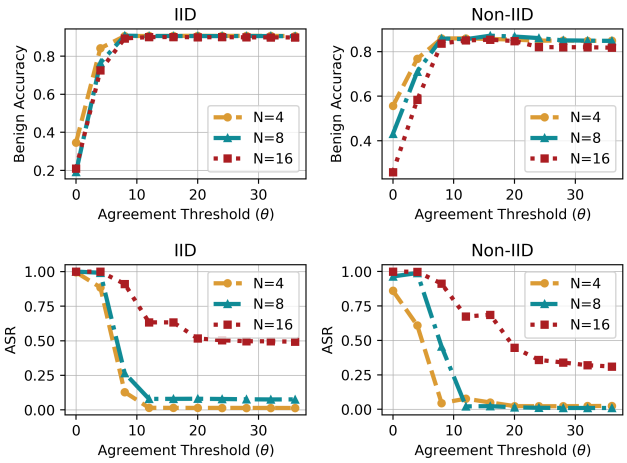

Figure 14. Impact of consensus fusion threshold of Lockdown in different # of attackers setting.