# A Supplementary Material

## A.1 Implementation Details

**Decay pruning rate with cosine annealing.** In our subspace pruning/recovery process, we let the clients prune out $\alpha_t$ percentage 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 $\alpha_0$ 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 $T_{end} = T$.

**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 $1\frac{n^{l-1}+n^l+w^l+h^l}{n^{l-1}*n^l*w^l*h^l}$, 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 $\mathrm{n}^l$ 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 2 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 from selecting them. After that, we filter out those parameters with the smallest $\alpha_t$ percentage of magnitude and prune them out of the subspace.

**Algorithm 2** PyTorch style code for pruning and recovery

1: **function Prune_subspace**( $\alpha_t$, $\boldsymbol{w}_{i,t,K}$, $\boldsymbol{m}_{i,t+\frac{1}{2}}$ )
2: Init layer sparsity $\{s_l\}$ given overall sparsity $s$ with ERK
3: $\boldsymbol{m}_{i,t+1} = \boldsymbol{m}_{i,t+\frac{1}{2}}$
4: **for** $l = 0, 1, \ldots, L-1$ **do**
5: $num_{prune} = \alpha_t \times$ # of params in the l-th layer
6: $sort =$ torch.where($\boldsymbol{m}_{i,t}^{(l)} == 1$, torch.abs($\boldsymbol{w}_{i,t,K}^{(l)}$), $1000\times$torch.ones_like($\boldsymbol{w}_{i,t,K}^{(l)}$))
7: _, $idx =$ torch.sort($(sort)$.view$(-1)$)
8: $\boldsymbol{m}_{i,t+1}^{(l)}$.view$(-1)[idx[{:}\ num_{prune}]] = 0$
9: **end for**
10: Return $\boldsymbol{m}_{i,t+1}$
11: **end function**
12:
13: **function Recover_subspace**( $\alpha_t$, $\boldsymbol{w}_{i,t,0}$, $\boldsymbol{m}_{i,t}$ )
14: Derive gradient $\nabla f_i(\boldsymbol{w}_{i,t,0} \odot \boldsymbol{m}_{i,t})$ with one pass of local data
15: $\boldsymbol{m}_{i,t+\frac{1}{2}} = \boldsymbol{m}_{i,t}$
16: **for** $l = 0, 1, \ldots, L-1$ **do**
17: $num_{prune} = \alpha_t \times$ # of params in the l-th layer
18: $sort =$ torch.where($\boldsymbol{m}_{i,t+\frac{1}{2}}^{(l)} = 0$, torch.abs($\nabla f_i^{(l)}(\boldsymbol{w}_{i,t,0} \odot \boldsymbol{m}_{i,t})$), $-1000\times$torch.ones_like($\boldsymbol{w}_{i,t,K}^{(l)}$))
19: _, $idx =$ torch.sort($(sort)$.view$(-1)$, descending=True)
20: $\boldsymbol{m}_{i,t+\frac{1}{2}}^{(l)}$.view$(-1)[idx[{:}\ num_{prune}]] = 1$
21: **end for**
22: Return $\boldsymbol{m}_{i,t+\frac{1}{2}}$
23: **end function**

**Subspace recovery.** After pruning and before the next round training, 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

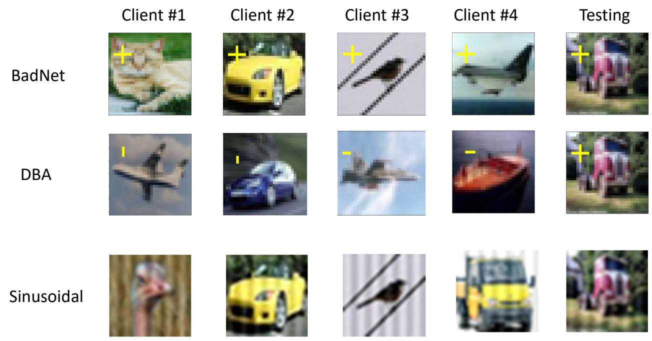


Figure 7: Examples of BadNet, DBA, and Sinusoidal attack. Labels of poison samples are manipulated to the target label (e.g., a horse).

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 Algorithm 2. Subsequently, we sort in descending to obtain the parameters with the largest-$\alpha_t$ percentage gradient magnitude, and recover them by updating masks.

### A.2 Attack Methods

Table 10: 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) | ✗ | ✓ | ✓ |
| 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 7.

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.

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. 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 0.25.
- **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 10 the attack methods we can perform with specific threat models.

### A.3 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 are able to launch adaptive attack 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 problem is solved by the Smoothed Weiszfeld Algorithm. Similar techniques are 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 $(\alpha, f)$-Byzantine resilience where $\alpha$ 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.

### A.4 Security Analysis

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 an adaptive attack 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 cannot 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.

### A.5 More Visualization

**Input-level visualization.** In Figure 8, 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.

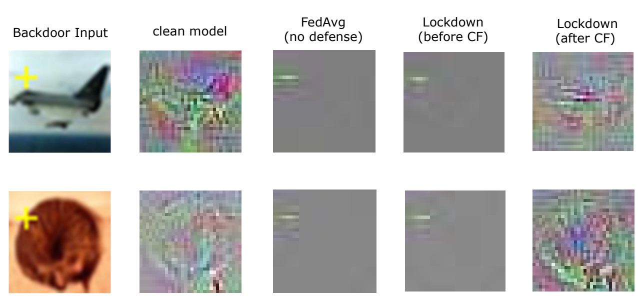


Figure 8: Smooth grad (Smilkov et al., 2017) visualization of models given backdoor input. The first column is data input with backdoor trigger. The subsequent columns demonstrate the gradient with respect to the input of i) a model without being poisoned. ii) a model trained by FedAvg with poisoned 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.

**Parameters-level visualization.** In Figure 9, 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).

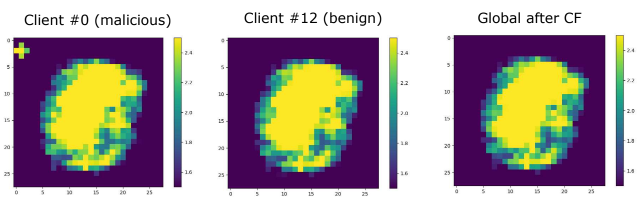


Figure 9: Visualization of absolute global weights after projecting into the local or global subspace. Left: projecting into local subspace of a malicious client. Middle: projecting into local subspace of a benign client. Right: projecting into consensus subspace produced by consensus fusion. The brighter 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.

## A.6 Ablation study

We perform ablation study of Lockdown on CIFAR10. BadNet is the default attack method.

**Gradient-based recovery vs. random recovery.** In subspace recovery process, 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 78.3% reduction) though the benign acc of the model suffered a little bit (by up-to 2.3% 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, 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 parameters (which essentially is what ERK does).

**Consensus fusion (CF).** In Figure 10, we demonstrate the necessity of 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 shows that masking out some malicious/dummy parameters can perturb the backdoor function and thereby curing the poisoned model.

Table 12: Ablation study for sparsity initialization.

| Methods (IID) | Benign Acc ↑ | ASR ↓ | Backdoor Acc ↑ |
|---|---|---|---|
| Uniform | 81.7 | **5.7** | 75.9 |
| ERK (ours) | **90.1** | 7.1 | **83.7** |

| Methods (Non-IID) | Benign Acc ↑ | ASR ↓ | Backdoor Acc ↑ |
|---|---|---|---|
| Uniform | 75.5 | **3.1** | 70.6 |
| ERK (ours) | **86.1** | 3.4 | **82.2** |

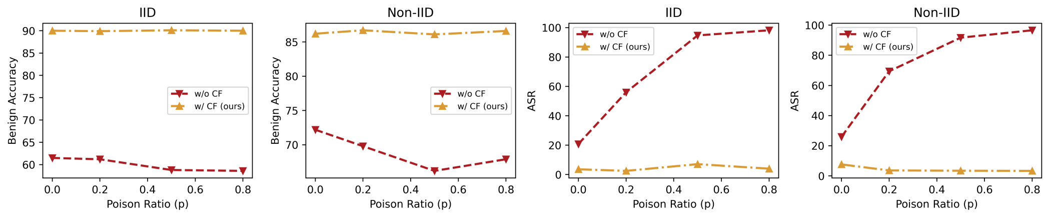

Figure 10: Impact of consensus fusion of Lockdown.

### A.7 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 2 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 understandable since Lockdown reduces to FedAvg when sparsity is 0. On the other hand, with larger sparsity, the benign accuracy of the model suffers due to the reduction of trainable parameters. Therefore, there exists a tradeoff for the sparsity of Lockdown. Larger sparsity promises lower model complexity, smaller comm 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 | **91.0** | 68.4 | 6.57M |
| 0.2 | 90.9 | 61.1 | 5.26M |
| 0.5 | **91.0** | 10.9 | 3.29M |
| 0.75 | 90.1 | 7.1 | 1.65M |
| 0.9 | 88.3 | **3.0** | **0.66M** |

| $s$ (Non-IID) | Benign Acc ↑ | ASR ↓ | # of params ↓ |
|---|---|---|---|
| 0 | **89.1** | 70.3 | 6.57M |
| 0.2 | 88.4 | 52.6 | 5.26M |
| 0.5 | 87.1 | 14.1 | 3.29M |
| 0.75 | 86.1 | 3.4 | 1.65M |
| 0.9 | 85.0 | **2.9** | **0.66M** |

**Initial pruning/recovery rate.** We also show the effect of initial pruning/recovery for the learning performance. As shown, larger pruning rate would typically results in the drop of benign accuracy but also enhance the ASR under poisoning attack. Specially, when $a_0 = 0$, lockdown reduces to train a sparse subnetowrk from scratch, without evolving the sparse coordinate. This setting cannot eliminate the "poison-couple" effect, therefore the ASR is as high as FedAvg with no defense. On the other hand, setting $\alpha_0$ will also result in isolation of subspace for different clients, resulting in lack of consensus in the global space and therefore leading to drop of benign accuracy.

**Consensus fusion threshold $\theta$.** In Figure 11, we tune the CF threshold $\theta$ to see its impact on different settings of attacker number $N$. In all settings of $N$, we see that: i) $\theta$ should not be set to be too small; otherwise, the benign accuracy would be lower, and the ASR will be higher. ii) $\theta$ also should not be set too large; otherwise, it will severely compromise benign accuracy, but the reduction

Table 14: Performance of Lockdown under different initial pruning/recovery rate $a_0$.

| $a_0$ (IID) | Benign Acc ↑ | ASR ↓ | Backdoor Acc ↑ |
|---|---|---|---|
| 0 | **90.5** | 49.4 | 47.7 |
| 1e-5 | 90.5 | 5.2 | 84.3 |
| 1e-4 | 90.1 | 7.1 | 83.7 |
| 1e-3 | 88.1 | 3.7 | **84.7** |
| 1e-2 | 87.2 | 3.5 | 83.7 |
| 1e-1 | 87.0 | **3.1** | 83.4 |

| $a_0$ (Non-IID) | Benign Acc ↑ | ASR ↓ | Backdoor Acc ↑ |
|---|---|---|---|
| 0 | **88.5** | 85.3 | 14.0 |
| 1e-5 | 87.4 | 8.5 | 78.8 |
| 1e-4 | 86.1 | 3.4 | **82.2** |
| 1e-3 | 84.9 | **2.1** | 80.4 |
| 1e-2 | 83.4 | 5.3 | 76.4 |
| 1e-1 | 83.7 | 5.2 | 77.6 |

of ASR will not be too significant. Per our results, the consensus 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, $\theta$ 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.

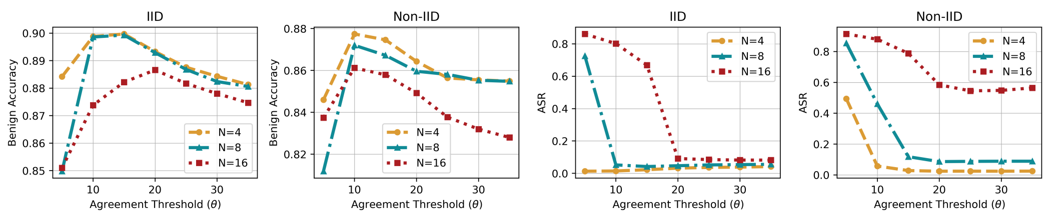


Figure 11: Impact of consensus fusion threshold of Lockdown in different # of attackers setting.

## A.8 Limitations

Our method utilizes sparsity of model to counter backdoor attack. However, we are aware that sparsity in its current stage can hardly guarantee acceleration of the training/inference speed. At present, the current sparse acceleration technique requires 2:4 sparse operation. More specifically, the 2:4 sparse operation requires that there are at most two non-zero values in four contiguous memory, which may not hold for the sparse model produced by Lockdown. But we insist that our method has great potential to achieve truly training acceleration with development of sparse technique.

There are potentially other adaptive backdoor attacks that can break the defense of lockdown, especially under the assumption that attackers have full control over its local training process and has knowledge of the defense. We leave the research of potential attacks against Lockdown as future works.

## A.9 Broader Impact

The poison-coupling effect we discover in this paper might be mis-used to guide the design of backdoor attack method in centralized learning/FL scenario. We will continue this line of research and further propose attack/defense method to better study/mitigate such an effect. We also open-source our code to facilitate researchers/machine learning engineer in academy/industry to study and understand the discovered phenomenon.