FedWrapper: A Plug-and-Play Federated Learning Module for Secure and Efficient Aggregation

Fangzhou Wu [1] **Chulin Xie** [2] **Yezhou Yang** [3] **Jia Zou** [3] **Chaowei Xiao** [1]
[1] University of Wisconsin–Madison, [2] UIUC, [3] ASU

Abstract

Federated Learning (FL) provides an effective collaborative training paradigm, allowing clients jointly to train a global model without sharing their local data. Despite being effective, it faces challenges in scenarios with data heterogeneity and is vulnerable to training-time attacks. To address them, in this work, we propose *FedWrapper*, a plug-and-play module that can be integrated seamlessly into existing FL frameworks. *FedWrapper* addresses data heterogeneity and improves the model robustness against training-time attacks by mixing the model weights of different clients via learned parameters in a self-supervised learning manner. Theoretically, we provide the convergence guarantees under Non-IID settings and explain why our method is robust against training time attacks. Empirically, we show that *FedWrapper* can plug into the existing FL frameworks and consistently improves both accuracy and robustness on CIFAR10 and CIFAR100.

1 Introduction

Federated Learning (FL) [23] is an effective distributed learning paradigm that leverages diverse data on different clients to train a global model without directly sharing their private data. However, standard FL algorithms, such as FedAvg [23], face ineffectiveness when dealing with data heterogeneity and are vulnerable to training-time attacks [6, 5, 27, 4, 43], challenging their practical deployment.

To tackle *data heterogeneity*, recent works [23, 21, 11, 51, 16, 40] have focused on improving training objective functions and reducing the gap between local updates and the global model (see Section 2 for more details). On the other hand, to enhance *robustness against malicious clients*, robust aggregation techniques [6, 47, 29] have been developed. These methods either identify and downscale weights of malicious updates during aggregation [6, 47] or rely on robust estimation of a true "center" of model updates such as geometric medians[29] or trimmed mean [47] (see Section 2). While effective, these existing approaches tend to address the data heterogeneity and training-time attacks separately.

There are a few works that address data heterogeneity and detect malicious clients at the same time. However, they often require additional local data shared from clients as a proxy dataset for server-side aggregation [44, 30, 7, 45]. Such data sharing is prohibited by today's regulatory compliances [3, 39, 34], and is thus impractical in real-world deployment. In addition, existing methods addressing data heterogeneity or robustness against training-time attacks have been designed independently and might not be readily compatible with each other because of the inconsistency in their objectives, assumptions, interfaces, and dependencies. Integrating these methods not only requires theoretical analysis, but also requires ensuring their modularity[17], e.g., unified objectives/assumptions, plug-and-play interfaces, and minimal external dependencies, allowing them to be seamlessly combined and applied within the FL framework.

As to our knowledge, there lacks a practical solution that can effectively handle both challenges simultaneously, which is the target problem that we want to address in this work.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

To address the problem, in this paper, we propose, *FedWrapper*, a plug-and-play aggregation module that can be integrated seamlessly into existing standard (horizontal) FL frameworks that contain the center sever aggregation process, without additional local data shared from clients. Unlike traditional methods that simply average the updates from local models, *FedWrapper* designs a *layer-wise aggregation module* to aggregate the layer-wise weights from different clients by using learnable parameters to calculate the global model. Such a design can effectively combine the strengths of different models, enhancing the global model's generalization against Non-IID data. To optimize the learnable aggregation parameters, we leverage a contrastive loss [41] with an unlabeled public dataset. To improve the robustness against training time attacks, *FedWrapper* can be naturally and easily refined with a *malicious filter module*, which uses learnable aggregation parameters to detect and filter out malicious local updates, without knowing detailed information about potential attacks or additional data from clients [45, 44, 30, 7].

We summarize our main contributions as follows:

- We propose a novel plug-and-play module *FedWrapper*, that can be integrated into the existing federated learning pipeline, to address the data heterogeneity challenge and improve the model robustness against malicious clients.
- We theoretically demonstrate the effectiveness of *FedWrapper* under non-IID settings by providing the convergence guarantee. Furthermore, we theoretically explain why *FedWrapper* is robust against training-time attacks.
- Empirically, we combine *FedWrapper* with the state-of-art standard federate learning frameworks that contain a center server aggregation including *FedDyn* and *FedProx*. We evaluate our framework on CIFAR10 and CIFAR100. We show that *FedWrapper* improves the accuracy by $0.5 \sim 1\%$ on the Non-IID setting. Furthermore, when confronted with training-time attacks, our method exhibits significantly superior robustness against strong attacks compared to the state-of-the-art defense methods including *RFA* [29], *Krum* [6], *Multi-Krum* [6], *Trimmed-Mean* [47].

2 Related Work

FL under heterogeneous data *FedAvg* [23] is a standard FL method that averages clients' local updates to obtain the global model at each communication round. However, due to the heterogeneous local data distributions among clients, the local updates would drift away from each other, making the aggregated model hard to converge [22]. A number of FL algorithms have been developed to address the challenge of data heterogeneity. For example, *FedProx* [21] adds a regularization term to the loss function to control the direction of local model updating and make the local models close to a global one. *FedDyn* [11] proposes a novel dynamic regularization method to ensure that the client optimum is asymptotically consistent with stationary points of the overall empirical loss thus keeping a relatively efficient and stable training process with a relatively low cost. *FedAVGM* [16] uses server momentum to mitigate unbalanced distributions. *FedMA*[40] matches and averages hidden elements that have similar feature signatures. *FedAvg-Share* [51] shares a small subset of client data. In addition, there also emerge other techniques such as model distillation [13] and leveraging the transformer architecture [32].

Robustness in FL Robust aggregation methods have been proposed to deal with the potential threats from adversarial clients. *Krum* and *Multi-Krum* [6] leverage pair-wise distance between the updates to detect abnormal clients. *Trimmed Mean* [47] proposes a new gradient decent algorithm based on medium and trimmed mean operations to achieve optimal statistical performance with Byzantine failures. *RFA* [29] completes the aggregation by computing the geometric median of the updates and an alternating minimization strategy to gain the most stable global model. *FABA* [42] removes one possible malicious client that has the largest difference with the mean gradients. *Diverse-FL* [30] uses a part of local training data to detect the poisoned updates. *Zeno* [44] utilizes a validation dataset on the server and computes the stochastic descendant score to eliminate the malevolent updates.

Contrastive learning Contrastive learning [8, 9, 14, 26] has gained significant success in the self-supervised learning area to deal with unlabeled data. The contrastive loss compares the similarity between sample pairs in the feature space and is designed to have a low value when sample x is

similar to its positive sample and dissimilar to the other samples (considered negative to x). Existing works [8, 14] commonly use the infoNCE [28] and NT-Xent [8] loss to measure this similarity and the loss is often in the form of entropy: $L_x = -\log \frac{\exp x \cdot x_{posi}}{\sum_{k=1}^{N} \exp x \cdot x_{k,neg}}$. Contrastive learning has been applied to federated learning for different purposes [20, 49, 52, 31, 38] from our work. For example, *FedCA* [49] and *FedU* [52] both use contrastive learning at client-side for unsupervised representation learning, and Moon [20] conducts contrastive learning at the model-level.

3 Method

Preliminaries and notations. At every communication round of standard FL, n participants are selected from a total of K clients. We denote the selected client set as S where $|S| = n$. FL aims to output a global model θ: $F(\cdot|\theta) = \sum_{j=1}^{n} p_j F_j(\cdot|\theta)$, where $F_j(\cdot|\theta) = \mathbb{E}_{\zeta_j \sim D_j}(f_j(\theta; \zeta_j))$ is j-th client's local empirical risk on its local private data D_j and $\sum_{j=1}^{n} p_j = 1$. For each client, they will share either gradients(e.g., FedSGD [24]) or parameters(e.g., FedAvg [23]) to calculate the global model. In this work, we present our method in the form of parameters and we provide convergence guarantees for both. For simplification, we introduce our framework based on FedAvg. Note that our framework can be easily extended to other federated aggregation mechanisms such as FedSGD, FedDyn [11], and FedProx [21] as shown in the supplementary material.

3.1 Overview

As shown in Figure 1, *FedWrapper* is designed as a plug-and-play module that can be easily integrated with standard FL algorithms that contain the center aggregation operation. *FedWrapper* consists of two components: (1) layer-wise aggregation module and (2) malicious filter module. After each communication round t, clients send their updated local models to the server. On the server side, *FedWrapper* first uses the malicious filter module to eliminate malicious clients' (as represented in red color in the figure) and the remaining benign clients are then fed into our novel layer-wise aggregation module, where layer-wise weights have been aggregated with learnable parameters, to obtain the global model. Finally, the server returns the global model to the clients for the next round (i.e., $t+1$) of updating.

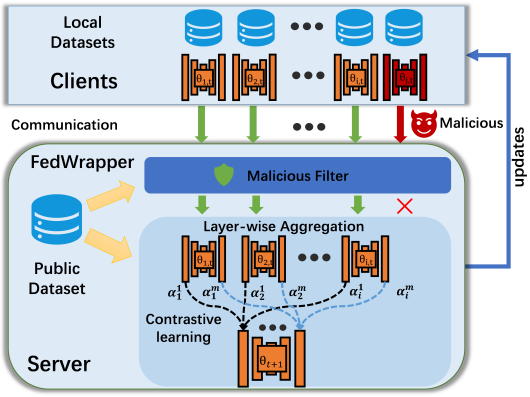


Figure 1: An illustration of *FedWrapper* module in round t.

3.2 Layer-wise aggregation

Different from the standard FedAvg, *FedWrapper* performs the model aggregation via layer-wise learnable parameters.

We denote the global model's i-th layer parameters at round t as $\theta^i_{\text{global},t}$ and the j-th local client's i-th layer parameter at round t as $\theta^i_{j,t}$. We can formalize the server aggregation process as: $\theta^i_{\text{global},t} = \sum_j^n \alpha^i_j * \theta^i_{j,t}$, where α^i_j is the i-th layer learnable coefficient for j-th client, w.r.t $\sum_{j=1}^{n} \alpha^i_j = 1$ and α^i_j are initialized as $\frac{1}{n}$.

To optimize the learnable parameters for better performance, in our settings, we leverage a small public dataset D_{pub}. Ideally, D_{pub} contains the data x and label y and follows a similar distribution with the clients' data such that supervised learning can be directly applied to optimize the learnable coefficients. However, labeled data is hard to acquire in real practice, due to various constraints [33]. Moreover, directly performing supervised training on D_{pub} makes the global model sensitive or overfitting to D_{pub}. In such a way, if D_{pub} is not similar to the distribution of the client's data, it will result in a significant performance drop. To address this, instead of using the supervised learning strategy to learn the data and label mapping, we adopt a contrastive learning strategy to optimize the

learnable parameters for learning the feature representation. Such a strategy not only removes the need for labels but also makes our module more robust and accurate.

In order to optimize the learnable coefficients, for simplification, given $x_i \in D_{\text{pub}}$, we first define the latent feature representation h_i of x as $h_i = F_{\alpha}^{\text{global}}(r(x_i))$, where F consists of different layers $\theta_0^{\text{global}} \circ \theta_1^{\text{global}} \circ \cdots \circ \theta_m^{\text{global}}$ and is parameterized by learnable coefficient $\alpha := \{\alpha_j^i |$ for all $i \in \{0, \cdots, m\}$ and $j \in \{0, \cdots, n\}\}$, r is the random augmentation [37]. For each data x_i, we randomly sample two augmentations, resulting in $h_i^1 = F_{\alpha}(r(x_i))$ and $h_i^2 = F_{\alpha}(r(x_i))$.

Since D_{pub} is the publically collected data, it may be long-tail distributed or highly correlated [41, 12]. If we directly use the NT-Xent loss [8], it may negatively impact the performance. Therefore, we leverage the cross-level discrimination (*CLD*) [41] as our objective function:

$$\alpha = \underset{\alpha}{\text{argmin}} \quad \underbrace{\{-\log \frac{\exp(\text{sim}(h_i^1, h_i^2))}{\sum_{k=1}^{B} \mathbb{1}_{k \neq i} \sum_{p=1}^{2} (\exp(\text{sim}(h_k^p, h_i^p)))}}_{\text{NT-Xent loss}} + \underbrace{\beta \frac{1}{2} \sum_{i=1}^{B} \sum_{p=1}^{2} \text{dist}(\text{centroids}(h_i^p), h_i^p)\}}_{\text{cross-level discrimination}} \tag{1}$$

where B is the batch size, $\text{sim}(h_k^p, h_i^p)$ is similarity metric of the feature representations, and $\text{centroids}(h_i)$ is the centroid of group where h_i belongs to. We leverage *K-Means* [46, 18] clustering to obtain the centroids.

3.3 Mailicous filter

To ensure a more robust global model, it is important to consider potential security threats that may disrupt the entire training process. The existence of Byzantine adversaries [6] is a prevalent concern in FL security research where the corrupted model parameters sent from adversarial clients can significantly deviate the FL global model from its optimum or even lead it in the wrong direction. To address this, we introduce a malicious filter component to improve the robustness of our method.

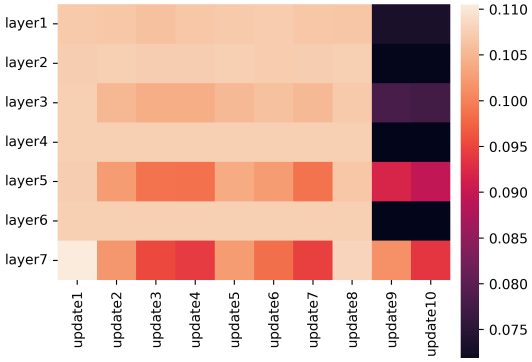


Figure 2: The heatmap of learnable coefficient α.

Since the malicious clients aim to downgrade the global model's performance, the model weights among different layers of the malicious clients will introduce a negative impact on the global model. Thus, we hypothesize that when optimizing the learnable coefficient α using the first part of equation 1 (i.e, the NT-Xent loss), the layer-wise coefficient α for the malicious client will be smaller than the benign ones. To verify our hypothesis, we show the heatmap of α in Figure 2. The last two columns are the weights among different layers for malicious agents. We can observe that their values are lower than others on average. Moreover, we also observe that it is not enough to only use the last layer to filter out malicious clients instead of using the average value among all layers. It further demonstrates the effectiveness of our design for robustly filtering out malicious clients. Based on it, we propose our malicious filter algorithm as follows: (1) we first define the counting matrix M, where $M_{i,j}$ represent the counter value for client j at i-th layer, and initialize all values as 0 ; (2) we optimize the NT-Xent loss (the first part of objective function) shown in equation 1 among all clients to calculate α; (3) For each layer i, we rank the α_j^i with respect to all clients j and store the client index p with the K lowest value in set P; We then set $M_{i,p} = M_{i,p} + 1$ for all $p \in P$; (4) we aggregate M along the layer dimension and filter out the clients that are associated with top-K values in M.

4 Theoretical analysis

4.1 Robustness

In this section, we analyze the robustness of the malicious filter. As for our algorithm, we use *SimCLR* [8] as the contrastive learning method. To compute the NT-Xent loss for a pair of positive examples,

we have:

$$l_i = -\log \frac{\exp(\text{sim}(h_i^1, h_i^2))}{\sum_{k=1}^{B} \mathbb{1}_{k \neq i} \sum_{p=1}^{2} (\exp(\text{sim}(h_k^p, h_i^p)))} \tag{2}$$

where the $\mathbb{1}$ is an indicator, and h_i^1, h_i^2 as the latent representation of a pair of positive examples generated by random augmentation [37].

We now introduce a basic assumption below:

Assumption 4.1 (Clustering effect)**.** *For a benign model θ_{ben} and a poisoned model θ_{poi}, given an unlabeled dataset D where the data has m latent classes noted as $c_i (i = 1, 2, \ldots, m)$, then the latent representations of data point x_i generated by these two models are marked as h_i^{ben} and h_i^{poi} and we have $h_i^{ben} \in c_i$. Assume $h_i^{ben} \sim P_{c_i}$ has deviation σ_{c_i}, $h_{c_{m/i}}^{ben} \sim P_{c_{m/i}}$ has deviation $\sigma_{c_{m/i}}$ and $h_i^{poi} \sim P_{all}$ has deviation σ_{all}. Formally we have $||\sigma_{c_j}|| < ||\sigma_{m/i}|| < ||\sigma_{all}||$. where $c_{m/i}$ means all latent classes except class c_i.*

The benign model can cluster the latent representations for m latent classes, and if the model parameters get poisoned by the adversary, the structure of the representation for the dataset D will be damaged (e.g.,the model parameters are perturbed by noise), the distribution can be assumed as a relatively uniformly situation and the representation h_D^{poi} can be seen as obeying the whole distribution P_{all}. The assumption is that the deviation of the benign model representation of a latent class is smaller than that of the rest classes and also the poisoned one.

Based on Assumption 4.1, we have Theorem 4.2 as follows.

Theorem 4.2. *if Assumption 4.1 is satisfied, we have*

$$\mathbb{E}{l_i}^{ben} = \log((2N-1)(1+\delta)) < \mathbb{E}{l_i}^{poi} = \log((2N-1))$$

where $\mathbb{E}{l_i}^{ben}$ and $\mathbb{E}{l_i}^{poi}$ represent the expected loss of benign updates and malicious updates respectively and $\delta < 0$.

This theorem demonstrates that contrastive loss can be a great metric to detect malicious updates to some extent (i.e., the NT-Xent loss is relatively lower than that in the benign model). In addition, the weights of these poisonous updates can be relatively lower than the benign ones in the wrapper model. The full proof can refer to Section 1.1 in the supplementary material.

4.2 Convergence

There are two basic paradigms in FL: updating the gradients or updating the model parameters [23]. Here we give the analysis of our module based on both updates.

Convergence based on FedAvg One basic form of our algorithm is based on the *FedAvg* (i.e., updates are model parameters). The main difference between our algorithm and the *FedAvg* is that *FedWrapper* fine-tunes learnable parameters layer-by-layer at the server side. Following [50], [36], [35], [22] and [48], we can formulate the following assumptions, where F_k is the loss function for k-th client.

Assumption 4.3. *(1) The loss function $F, F_1, \cdots, F_N$ are all L-smooth and μ-strongly-convex. That is, for all w and v: $F_k(w) + (v-w)\nabla F_k(w) + \frac{\mu}{2}||w-v||_2^2 \leq F_k(v) \leq F_k(w) + (v-w)\nabla F_k(w) + \frac{L}{2}||w-v||_2^2$;*

(2) Let ϵ_t^k be the data sample drawn from the k-th device's local data uniformly at random. The variance of gradients in each client is bounded: $\mathbb{E}||\nabla F_k(w_k^t, \epsilon_k^t) - \nabla F_k(w_k^t)||^2 \leq \sigma_k^2$ for $k = 1, \cdots, N$.

(3) for $k = 1, \cdots, N$ and $t = 1, \cdots, T-1$, the expected l_2 norm of gradients is bounded: $\mathbb{E}||\nabla F_k(w_k^t, \epsilon_k^t)||^2 \leq G^2$

Here, we denote w as the global training model and w_k^t as the k-th local model at round t. Then we have the following theorem Theorem 4.4.

Theorem 4.4. *Let Assumption 4.3 holds, the local training epoch is E_{local} and for any t, learning rate satisfies $\eta_{t_1} \leq E_{local}\eta_{t_1+E_{local}} \leq E_{local}\eta_t$ for $t_1 \leq t \leq E_{local} + t_1$ and $\sum_t \eta_t^2 = I < \infty$ then,*

$$\mathbb{E}(F(w_t)) - F^\star \leq \frac{L}{2}IQ + \mathbb{E}||\overline{w}_0 - w^\star||^2$$

where we set $\tau = \sum_{k=1}^{N}\sum_{i=1}^{m}\alpha_k^i(F_k(w_0^k) - F_k^\star)$ and $Q = m\sum_{k=1}^{N}\sum_{i=1}^{m}\alpha_k^i\sigma_k^2 + 2L(m+1)\tau + (E_{local})^2 m(E_{local}-1)^2G^2$.

The detailed proof for Theorem 4.4 is deferred to Section 1.2 in supplementary material.

Convergence based on FedSGD In gradients updating situation, for convenience, we denote the process of the training as the following sequence:

$$w_{t+1} = w_t - \eta_l FedWrapper(\nabla l(w_t, \zeta_t^k)) \tag{3}$$

Here, we define w_t as the global model at round t, η_l as the local learning rate ζ_t^k as the training mini-batch in k-th client. Now we will give one basic assumption to prove the convergence in Theorem 4.6.

Assumption 4.5. *(a) The loss function $L(w)$ is three times differentiable with continuous derivatives and L(w) $\geq$ 0. (b) the learning rate satisfies $\sum \eta_t^2 < \infty$ and $\sum \eta_t = \infty$ (c) we set the idea gradient expectation as $\mathbb{E}G(w) = \nabla L(w)$ the moments $\mathbb{E}||\nabla L(w)_c||^q \leq A_q + B_q||w||^q$, for $q = 2, 3, 4$. (d) Assume that there exists a horizon D, we have,*

$$\inf_{w^2 \geq D} -w\mathbb{E}\nabla L(w)_c > 0$$

Then, we derive the following Theorem 4.6.

Theorem 4.6. *if FedWrapper satisfies Assumption 4.5, and let $0 \leq \alpha < \frac{\pi}{2}$ be any angular value. Let $w_1, \ldots, w_n$ be any independent identically distributed random vectors in $\mathbb{R}^d$, $w_i \sim G$, with $\mathbb{E}G = g$. We denote S as FedWrapper($w^{\{n\}}$). Then it satisfies*

(i) $\langle \mathbb{E}S, g\rangle \geq (1 - \sin\alpha) \cdot ||g||^2$

(ii) for $r = 2, 3, 4$, the moments $\mathbb{E}||S||^r$ is bounded by a linear combination of terms $\mathbb{E}||G||^{r_1} \ldots \mathbb{E}||G||^{r_{n-1}}$ with $r_1 + \cdots + r_{n-1} = r$.

Then the loss function $L(w)$ is convergent.

The specific proof for Theorem 4.6 can be seen in Section 1.3 in the supplementary material.

5 Experiments

In this section, we first introduce our experiment settings in Section 5.1, and present the evaluation results of *FedWrapper* under Non-IID settings without attacks in Section 5.2. Then we verify the robustness of *FedWrapper* against training-time attacks in Section 5.3. Finally, we perform various ablation studies to provide insights into our method in Section 5.4.

5.1 Settings

Dataset and network. We mainly use two datasets for evaluation: CIFAR10 [19] and CIFAR100 [19]. For the server-side unlabeled public data, we choose CIFAR100 as the unlabeled data for CIFAR10 and vice versa. For the classifier, we choose two widely used architectures: ResNet-18 [15] and CNN [2] for CIFAR10 and CIFAR100 following the *FedDyn* experiments [11]. As *FedWrapper* can be seen as the plug-and-play module, we test three main frameworks to combine with: the most basic method *FedAvg* and the state-of-art frameworks: *FedDyn* and *FedProx*. For detailed experimental settings, please refer to Section 2.1 in supplementary material.

Training details In our aggregation process, we use 2000 unlabeled public data to optimize the learnable coefficient parameters and set optimizing epochs= 2. For the setting of Federated Learning, we follow the standard setting in [11]. we set 10 (10% of 100) clients to participate in training each

round. We set 1000 training rounds for CIFAR10/CIFAR100. In addition, We set 1500 training rounds for *FedAvg* on CIFAR100.

Non-IID settings. To simulate data heterogeneity in real operation, we follow the standard setting and use Dirichlet distribution [16, 25] to split the training data where the parameter γ controls the degree of dispersion. In our experiment, we set two different Non-IID degrees: $\gamma = 0.6$, $\gamma = 0.3$. .

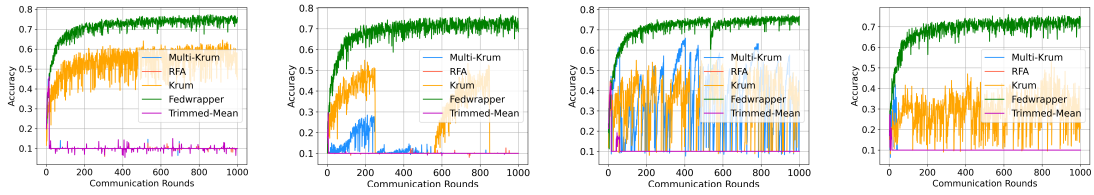
Attacks settings. We select two model parameter attacks: 1) layer-wise Gaussian noise attack, where we add different Gaussian noise among different layers. 2) layer-wise sign-flipping attack [10], where the changes of signs are different among different layers. For layer-wise Gaussian noise setting, We set μ^t as the mean of the t-th layer noise which is uniformly sampled from $(0, 0.1)$ and σ^t as the deviation of the t-th layer noise which is uniformly sampled from $(0, 0.05)$. For layer-wise sign-flipping attack, we uniformly sample the control parameter β^t in t-th layer w^t from $(-1.0, -0.5)$, thus the poisoned layer t in model w becomes $w^t_{poi} = w^t \cdot \beta^t$. For varying attack strengths, we poison different proportions of clients, namely 5%, 10%, 15%, 20%, and 25%. In the meantime, we select the state-of-art robust aggregations as baselines including *RFA*, *Krum* [6], *Multi-Krum* [6], *Trimmed-Mean* [47]. To facilitate effective comparisons with existing methods that cannot handle more than 50% malicious clients, such as *Krum* and *Multi-Krum*, we limit the poisoned clients in each round to a maximum of 50%. We set the $filter\_num$ as 70% for our algorithm and evaluate the impact of this number in the ablation study. For *Trimmed-Mean*, we also discard 70% clients in each round.

5.2 Non-IID results

In this section, we evaluate the performance under Non-IID scenarios both with and without *FedWrapper*. We do not add the malicious filter module in this setting for fair comparison with baselines.

Table 1: Accuracy for *FedWrapper* module combined with FedAvg/FedDyn/FedProx.

Algorithm	Max_acc			
	CIFAR10 (γ)		CIFAR100 (γ)	
	0.6	0.3	0.6	0.3
FedAvg	80.49%	79.69%	45.40%	43.79%
FedWrapper+FedAvg	**81.00%**	**80.02%**	**45.78%**	**44.16%**
FedDyn	82.85%	82.12%	48.42%	48.35%
FedWrapper+FedDyn	**83.06%**	**82.41%**	**49.07%**	**48.74%**
FedProx	80.50%	79.97%	45.14%	44.72%
FedWrapper+FedProx	**80.96%**	**80.22%**	**45.41%**	**45.06%**

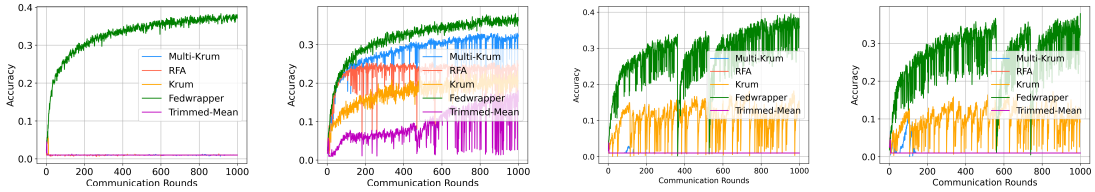


(a) layer-wise Gaussian noise attack. Set $\gamma = 0.6$. (b) layer-wise Gaussian noise attack. Set $\gamma = 0.3$. (c) layer-wise Sign-flip attack. Set $\gamma = 0.6$. (d) layer-wise Sign-flip attack. Set $\gamma = 0.3$.

Figure 3: The robustness against layer-wise Gaussian noise and layer-wise sign-flip attack. The training data is CIFAR10.

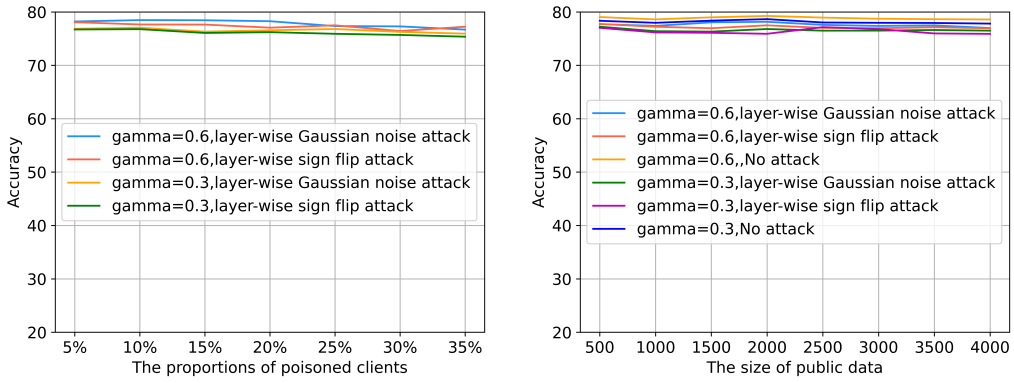
CIFAR10. For CIFAR10, the results are shown in Table 1. We find that by using *FedWrapper*, the model can achieve consistently better performance. In the setting $\gamma = 0.6$, our module can improve maximum accuracy by 0.51% when combined with *FedAvg*, 0.46% with *FedProx* and 0.21% when combined with *FedDyn*. Additionally, when setting $\gamma = 0.3$, it can improve 0.33%, 0.27% and 0.29% when combined with *FedAvg*, *FedProx* and *FedDyn* respectively.

CIFAR100. The results of CIFAR100 are shown in Table 1. We observe a similar improvement when combining with our method. From the table, we find *FedWrapper* can improve 0.38%, 0.27% and



(a) layer-wise Gaussian noise attack. Set $\gamma = 0.6$. (b) layer-wise Gaussian noise attack. Set $\gamma = 0.3$. (c) layer-wise Sign-flip attack. Set $\gamma = 0.6$. (d) layer-wise Sign-flip attack. Set $\gamma = 0.3$.

Figure 4: The robustness against layer-wise Gaussian noise and layer-wise sign-flip attack. The training data is CIFAR100.



(a) The performance against varying proportions of poisoned clients ($filter\_num = 7$). (b) The performance against various numbers of public data samples ($filter\_num = 7$).

Figure 5: Ablation study on the number of poisoned clients and size of public data .

0.65% respectively, when combined with *FedAvg*, *FedProx* and *FedDyn* using $\gamma = 0.6$, and 0.37%, 0.34% and 0.39% respectively, when using $\gamma = 0.3$.

5.3 Robustness against training-time attacks

We evaluate the robustness against two attacks: 1) layer-wise Gaussian noise. 2) layer-wise sign-flip attack. The results are shown in Figure 3 and Figure 4. We can observe that the performance with 25% clients poisoned and $\gamma = 0.6/0.3$ on both CIFAR10 and CIFAR100. The rest of the results are in the supplementary materials.

The results show that even with a relatively high degree of Non-IID and a high percentage of malicious clients, our method achieves significant performance improvements compared to baselines. It verifies the effectiveness of our method against these two training-time attacks.

Table 2: Impact of robustness parameter $filter\_num$ with $\gamma = 0.6$ and poisoned clients as 25%. We define $filter\_num$ as fn.

Dataset	Attack	fn=5	fn=6	fn=7	fn=8	fn=9
CIFAR10	layer-wise Gaussion noise attack	**79.17%**	78.36%	77.35%	77.12%	72.85%
	layer-wise sign-flip attack	77.05%	77.38%	**77.49%**	76.75%	72.7%
CIFAR100	layer-wise Gaussion noise attack	30.31%	28.01%	**38.32%**	36.99%	32.05%
	layer-wise sign-flip attack	23.61%	26.92%	**38.53%**	35.31%	32.15%

5.4 Ablation studies

Robustness filter parameter. We first show how the robustness filter parameter $fliter\_num$ affects the robustness performance of our module based on *FedAvg*. We set poisoned clients as 25% and change the $fliter\_num$. We calculate the maximum robust accuracy among 1000 communication rounds as a metric. The results are shown in Table 2. We can observe that if the filter number is not too large, the robust accuracy will increase with $fliter\_num$ at first. We guess the reason is that when increasing the $fliter\_num$ it is more likely to detect the malicious updates. However, if the filter number is too large, the robust accuracy decreases a bit. The reasons are that in such a way, our method filters out too much data including the data from the benign client agents so that the rest data is not enough to train a good model. Notably, the optimal $fliter\_num$ for the best robust accuracy is near 70% of the number of updates for both attacks.

We further fix the $filter\_num$ to 7 and change the malicious clients' ratio in a range from 5% to 35%. We evaluate the performance against layer-wise Gaussian noise and layer-wise sign-flip attacks with $\gamma = 0.6/0.3$ using CIFAR10 as training data and CIFAR100 as server public data. The results are illustrated in Figure 5 (a). The results indicate that *FedWrapper* can effectively mitigate the effects of poisoning attacks, even when up to 35% of clients are malicious.

Table 3: The impact of the distribution of data on the server.

Server data	Acc.(No attack)	Acc.(Noise)	Acc.(Flip)
2000 samples from CIFAR10	81.04%	77.54%	77.61%
2000 samples from CIFAR100	81.00%	77.35	77.49%
2000 samples from STL10	80.91%	77.29%	77.32%
Without *FedWrapper* (FedAvg)	80.49%	10%	10%

Server datasize & data distribution. As *FedWrapper* needs unlabeled data on the server side, we investigate the impact of tuning the total amount of unlabeled data and its corresponding distribution. First, we use CIFAR10 as training data and CIFAR100 as server data with the size varying in a range from 500 to 4000 samples. We conducted the experiments under three different scenarios: 1) no attack, 2) layer-wise Gaussian noise attack, and 3) layer-wise sign-flip attack. The results are presented in Figure 5 (b). The figure above illustrates that our method is effective regardless of the data size. With even a small amount of data (500 samples), our approach maintains a promising performance. Second, to investigate the impact of data distribution, we tested the performance of three different server datasets (training data remains to be CIFAR10), each containing 2000 uniformly selected samples from CIFAR100, STL10 [1], and CIFAR10, respectively. We evaluate the performance of these three different server datasets under three situations with $\gamma = 0.6$: 1) no attack, 2) layer-wise Gaussian noise attack, and 3) layer-wise sign-flip attack. The results are presented in Table 3, which revealed that the distribution of server data has a negligible effect on the results.

6 Conclusions and Limitations

Conclusions. We proposed a novel plug-and-play FL module called *FedWrapper*. The module incorporates local updates into a single, trainable wrapper model, fine-tunes the weights of each layer of the updates, and filters out malicious updates. In addition, we have applied contrastive learning to remove the need for data labels on the server and to improve the convergence and robustness, even with small amounts of publicly available unlabeled data. To demonstrate the state-of-the-art performance and robustness of our module, we integrated our method with existing state-of-art Federated Learning methods and conducted extensive experiments among different non-IID settings on CIFAR10 and CIFAR100 using different networks. As a result, we found that our method can achieve higher accuracy compared with baselines. Furthermore, our module can act as an effective filter against training-time attacks and achieve significantly superior robustness compared to the state-of-art defense method, *RFA* without any knowledge about client data and potential attacks.

Limitations. One problem with our method is that we need to optimize the objective twice. The first time aims to identify the malicious clients and the second time aims to update learnable parameters. It will introduce a training time overhead. We will leave the one-step optimization as the future work.

Acknowledgement We thank the support of the U.S. Department of Homeland Security under Grant Award Number, 17STQAC00001-07-00. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security

References

[1] Stl dataset. `https://cs.stanford.edu/~acoates/stl10/`.

[2] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (ICET)*, pages 1–6. Ieee, 2017.

[3] George J Annas. Hipaa regulations: a new era of medical-record privacy? *New England Journal of Medicine*, 348:1486, 2003.

[4] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 2938–2948. PMLR, 2020.

[5] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. In *International Conference on Machine Learning*, pages 634–643. PMLR, 2019.

[6] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. *Advances in Neural Information Processing Systems*, 30, 2017.

[7] Xiaoyu Cao, Minghong Fang, Jia Liu, and Neil Zhenqiang Gong. Fltrust: Byzantine-robust federated learning via trust bootstrapping. In *ISOC Network and Distributed System Security Symposium (NDSS)*, 2021.

[8] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[9] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. Big self-supervised models are strong semi-supervised learners. *Advances in neural information processing systems*, 33:22243–22255, 2020.

[10] Weilun Chen, Zhaoxiang Zhang, Xiaolin Hu, and Baoyuan Wu. Boosting decision-based black-box adversarial attacks with random sign flip. In *European Conference on Computer Vision*, pages 276–293. Springer, 2020.

[11] Alp Emre Durmus, Zhao Yue, Matas Ramon, Mattina Matthew, Whatmough Paul, and Saligrama Venkatesh. Federated learning based on dynamic regularization. In *International Conference on Learning Representations*, 2021.

[12] Adam R Ferguson, Jessica L Nielson, Melissa H Cragin, Anita E Bandrowski, and Maryann E Martone. Big data from small data: data-sharing in the'long tail'of neuroscience. *Nature neuroscience*, 17(11):1442–1447, 2014.

[13] Xuan Gong, Abhishek Sharma, Srikrishna Karanam, Ziyan Wu, Terrence Chen, David Doermann, and Arun Innanje. Ensemble attention distillation for privacy-preserving federated learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15076–15086, 2021.

[14] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[16] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

[17] Peter Kriens and Tim Verbelen. What machine learning can learn from software modularity. *Computer*, 55(9):35–42, 2022.

[18] K Krishna and M Narasimha Murty. Genetic k-means algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 29(3):433–439, 1999.

[19] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[20] Qinbin Li, Bingsheng He, and Dawn Song. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10713–10722, 2021.

[21] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2:429–450, 2020.

[22] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. In *International Conference on Learning Representations*.

[23] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[24] H Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. Federated learning of deep networks using model averaging. *arXiv preprint arXiv:1602.05629*, 2, 2016.

[25] Thomas Minka. Estimating a dirichlet distribution, 2000.

[26] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6707–6717, 2020.

[27] Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C Lupu, and Fabio Roli. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 27–38, 2017.

[28] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[29] Krishna Pillutla, Sham M Kakade, and Zaid Harchaoui. Robust aggregation for federated learning. *IEEE Transactions on Signal Processing*, 70:1142–1154, 2022.

[30] Saurav Prakash and Amir Salman Avestimehr. Mitigating byzantine attacks in federated learning. *arXiv preprint arXiv:2010.07541*, 2020.

[31] Tao Qi, Fangzhao Wu, Chuhan Wu, Lingjuan Lyu, Tong Xu, Hao Liao, Zhongliang Yang, Yongfeng Huang, and Xing Xie. Fairvfl: A fair vertical federated learning framework with contrastive adversarial learning. *Advances in Neural Information Processing Systems*, 35:7852–7865, 2022.

[32] Liangqiong Qu, Yuyin Zhou, Paul Pu Liang, Yingda Xia, Feifei Wang, Ehsan Adeli, Li Fei-Fei, and Daniel Rubin. Rethinking architecture design for tackling data heterogeneity in federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10061–10071, 2022.

[33] Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. Snorkel: Rapid training data creation with weak supervision. *The VLDB Journal*, 29(2-3):709–730, 2020.

[34] Protection Regulation. General data protection regulation. *Intouch*, 25:1–5, 2018.

[35] Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified sgd with memory. *Advances in Neural Information Processing Systems*, 31, 2018.

[36] Sebastian Urban Stich. Local sgd converges fast and communicates little. In *ICLR 2019-International Conference on Learning Representations*, number CONF, 2019.

[37] Ryo Takahashi, Takashi Matsubara, and Kuniaki Uehara. Data augmentation using random image cropping and patching for deep cnns. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(9):2917–2931, 2019.

[38] Yue Tan, Guodong Long, Jie Ma, Lu Liu, Tianyi Zhou, and Jing Jiang. Federated learning from pre-trained models: A contrastive learning approach. *arXiv preprint arXiv:2209.10083*, 2022.

[39] Timothy M Virtue. *Payment card industry data security standard handbook*. Wiley Online Library, 2009.

[40] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. *arXiv preprint arXiv:2002.06440*, 2020.

[41] Xudong Wang, Ziwei Liu, and Stella X Yu. Unsupervised feature learning by cross-level instance-group discrimination. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12586–12595, 2021.

[42] Qi Xia, Zeyi Tao, Zijiang Hao, and Qun Li. Faba: an algorithm for fast aggregation against byzantine attacks in distributed neural networks. In *IJCAI*, 2019.

[43] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. Dba: Distributed backdoor attacks against federated learning. In *International conference on learning representations*.

[44] Cong Xie, Sanmi Koyejo, and Indranil Gupta. Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance. In *International Conference on Machine Learning*, pages 6893–6901. PMLR, 2019.

[45] Yueqi Xie, Weizhong Zhang, Renjie Pi, Fangzhao Wu, Qifeng Chen, Xing Xie, and Sunghun Kim. Optimizing server-side aggregation for robust federated learning via subspace training. *arXiv preprint arXiv:2211.05554*, 2022.

[46] Jieping Ye, Zheng Zhao, and Mingrui Wu. Discriminative k-means for clustering. *Advances in neural information processing systems*, 20, 2007.

[47] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning*, pages 5650–5659. PMLR, 2018.

[48] Hao Yu, Sen Yang, and Shenghuo Zhu. Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5693–5700, 2019.

[49] Fengda Zhang, Kun Kuang, Zhaoyang You, Tao Shen, Jun Xiao, Yin Zhang, Chao Wu, Yueting Zhuang, and Xiaolin Li. Federated unsupervised representation learning. *arXiv preprint arXiv:2010.08982*, 2020.

[50] Yuchen Zhang, Martin J Wainwright, and John C Duchi. Communication-efficient algorithms for statistical optimization. *Advances in neural information processing systems*, 25, 2012.

[51] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

[52] Weiming Zhuang, Xin Gan, Yonggang Wen, Shuai Zhang, and Shuai Yi. Collaborative unsupervised visual representation learning from decentralized data. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4912–4921, 2021.