# Handling Both Stragglers and Adversaries for Robust Federated Learning

**Jungwuk Park** [*1] **Dong-Jun Han** [*1] **Minseok Choi** [2] **Jaekyun Moon** [1]

## Abstract

While federated learning (FL) allows efficient model training with local data at edge devices, among major issues still to be resolved are: slow devices known as stragglers and malicious attacks launched by adversaries. While the presence of both of these issues raises serious concerns in practical FL systems, no known schemes or combinations of schemes effectively address them at the same time. We propose Sageflow, staleness-aware grouping with entropy-based filtering and loss-weighted averaging, to handle both stragglers and adversaries simultaneously. Model grouping and weighting according to staleness (arrival delay) provides robustness against stragglers, while entropy-based filtering and loss-weighted averaging, working in a complementary fashion at each grouping stage, counter a wide range of adversary attacks. Extensive experimental results show that Sageflow outperforms various existing methods aiming to handle stragglers/adversaries.

## 1. Introduction

Federated learning (FL) (McMahan et al., 2017; Konečnỳ et al., 2016) is a promising direction for large-scale learning, which enables training of a global model with less privacy concerns. However, among major issues that need to be addressed in current FL systems are the devices called stragglers that are considerably slower than the average and the adversaries that enforce a various form of attacks.

Regarding the first issue, simply waiting for all the stragglers can significantly slow down the training process of FL. To handle stragglers, various asynchronous FL schemes have been proposed in the literature (Li et al., 2019b; Xie et al., 2019a; van Dijk et al., 2020). Especially in FedAsync (Xie et al., 2019a), the global model is updated asynchronously according to the device's *staleness* $t - \tau$, the difference between the current round $t$ and the past round $\tau$ when the device first received the global model. While the asynchronous schemes are highly effective in handling stragglers, its one-by-one update scheme does not lend itself well for integration with established aggregation methods to combat the second issue, the adversaries.

There are different forms of adversarial attacks that degrade performance of current FL systems: untargeted attacks such as model-update poisoning (Blanchard et al., 2017) or data poisoning (Biggio et al., 2012; Liu et al., 2017), and targeted attacks (or backdoor attacks) (Chen et al., 2017a; Bagdasaryan et al., 2018). Robust federated averaging (RFA) of (Pillutla et al., 2019), a well-known method proposed to handle adversaries in FL, employs geometric-median-based aggregation to provide a fair level of protection. Other various aggregation schemes (e.g., Multi-Krum) also successfully handle adversaries in distributed learning (Blanchard et al., 2017; Chen et al., 2017b). Unfortunately, however, the performance of these methods are substantially degraded when the portion of adversaries is large. The presence of stragglers drives the attack ratio higher (e.g., by ignoring stragglers), significantly degrading the performance of current aggregation schemes.

While the presence of both stragglers and adversaries raises significant concerns in current FL, to our knowledge, there are currently no existing methods or combinations of ideas that can effectively handle these issues simultaneously.

**Main contributions.** We propose Sageflow, staleness-aware grouping with entropy-based filtering and loss-weighted averaging, a robust FL strategy which can handle both stragglers and adversaries at the same time. Targeting the straggler issue, our strategy is to perform periodic global aggregation while allowing the results sent from stragglers to be aggregated in later rounds. In each global round, we take advantage of the results sent from stragglers by first grouping the models that come from the same initial models (i.e., same staleness), to obtain a group representative model. Then, we aggregate the representative models of all groups based on their staleness, to obtain the global model. Our grouping strategy based on staleness is not only effective in neutralizing stragglers but also provides a great platform for

*Equal contributions. [1]School of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), South Korea. [2]Department of Telecommunication Engineering, Jeju National University, South Korea.

This work was presented at the International Workshop on Federated Learning for User Privacy and Data Confidentiality in Conjunction with ICML 2021 (FL-ICML'21). 

countering adversaries, as discussed below.

Targeting each grouping stage of our straggler-mitigating idea, we propose an intra-group defense strategy which is based on our entropy-based filtering and loss-weighted averaging. These two methods work in a highly complementary fashion to effectively counter a wide range of adversarial attacks in each grouping stage. Here, in computing the entropy and loss of each received model, we utilize *public data* that we assume to be available at the server. In fact, the utilization of public data is not a new idea, as seen in recent FL setups of (Zhao et al., 2018; Xie et al., 2019c; Li et al., 2020). This is generally a reasonable setup since data centers typically have some collected data that can be accessed by public. We show later via experiments that only a very small amount of public data is necessary at the server (2% of the entire dataset, which is comparable to the amount of local data at a single device) to successfully combat adversaries. Our main contributions are as follows:

- We propose **Sageflow**, handling both stragglers and adversaries simultaneously in FL, via a novel **staleness-aware grouping** combined with **entropy-based filtering and loss-weighted averaging**.
- We derive the **theoretical bound** for Sageflow and provide key insights into the convergence behavior.
- Experimental results show that Sageflow **outperforms** various combinations of straggler/adversary defense methods using only a small portion of public data.

**Related works.** The authors of (Li et al., 2019b; Wu et al., 2019; Xie et al., 2019a; Lu et al., 2020; van Dijk et al., 2020) have recently tackled the straggler issue in FL. The basic idea is to allow the devices and the server to update the models asynchronously; the global model is updated every time the server receives a model from a device. However, a critical issue is that grouping-based (or aggregation-based) methods designed to handle adversaries, such as RFA (Pillutla et al., 2019), Multi-Krum (Blanchard et al., 2017) or the presently proposed entropy/loss based idea, are hard to be implemented in conjunction with these schemes since the model update is performed one-by-one asynchronously. Compared to the asynchronous schemes, our staleness-aware grouping can be combined smoothly with various aggregation rules against adversaries; we can apply RFA, Multi-Krum or our entropy/loss based idea in each grouping stage to obtain the group representative model.

To combat adversaries, various aggregation methods have been proposed in a distributed learning setup with IID data across nodes (Yin et al., 2018a;b; Chen et al., 2017b; Blanchard et al., 2017). The authors of (Chen et al., 2017b) suggests a geometric-median-based aggregation rule of models or gradients. In Multi-Krum (Blanchard et al., 2017), among $N$ workers in the system, the server tolerates $f$ Byzantine workers, where $2f + 2 < N$. Targeting FL with non-IID (independent, identically distributed) data distribution, RFA of (Pillutla et al., 2019) utilizes the geometric median of models sent from devices, similar to (Chen et al., 2017b). However, as already implied, these methods are ineffective when combined with a straggler-mitigation scheme (e.g., ignoring stragglers), potentially degrading the performance of learning. Compared to Multi-Krum and RFA, our entropy/loss based scheme can tolerate adversaries even with a high attack ratio, showing remarkable advantages when combined with straggler-mitigation schemes.

Finally, we note that Zeno (Xie et al., 2019b) and Zeno+ (Xie et al., 2019c) also utilize public data at the server to handle adversaries, but in a distributed learning setup with IID data across the nodes. Compared to Zeno+, our Sageflow targets non-IID data distribution in a FL setup. While Zeno+ adopts a fully asynchronous update rule (without considering the staleness) with the loss-based score function, our Sageflow integrates staleness-aware grouping, a semi-synchronous straggler-handling method, with entropy filtering and loss-weighted averaging, a harmonized means to provide protection against a wider variety of attacks.

## 2. Proposed Sageflow for Federated Learning

We consider the following federated optimization problem:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} F(\mathbf{w}) = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{k=1}^{N} \frac{m_k}{m} F_k(\mathbf{w}), \quad (1)$$

where $N$ is the number of devices, $m_k$ is the number of data samples in device $k$, and $m = \sum_{k=1}^{N} m_k$. By letting $x_{k,j}$ be the $j$-th data sample in device $k$, the local loss function of device $k$, $F_k(\mathbf{w})$, is written as $F_k(\mathbf{w}) = \frac{1}{m_k} \sum_{j=1}^{m_k} \ell(\mathbf{w}; x_{k,j})$.

### 2.1. Staleness-Aware Grouping against Stragglers

In the $t$-th global round, the server sends the current model $\mathbf{w}_t$ to $K$ devices in $S_t$, a randomly selected subset of $N$ devices. We let $C = \frac{K}{N}$ be the ratio of devices that participate in each global round. Each device in $S_t$ performs $E$ local updates and sends the updated model back to the server.

In handling slow devices, our idea assumes periodic global aggregation at the server. At each global round $t$, the server transmits the current model and time stamp, $(\mathbf{w}_t, t)$, to the devices in $S_t$. Instead of waiting for all devices in $S_t$, the server aggregates the models that arrive by a fixed time deadline $T_d$ to obtain $\mathbf{w}_{t+1}$, and moves on to the next global round $t + 1$. Hence, model aggregation is performed periodically with every $T_d$. A key feature here is that we do not ignore the results sent from stragglers not arriving by the deadline $T_d$. These results are utilized at the next global aggregation step, or even later, depending on the delay or staleness. Let $U_i^{(t)}$ be the group of devices selected from the server at global round $t$ that successfully sent their results

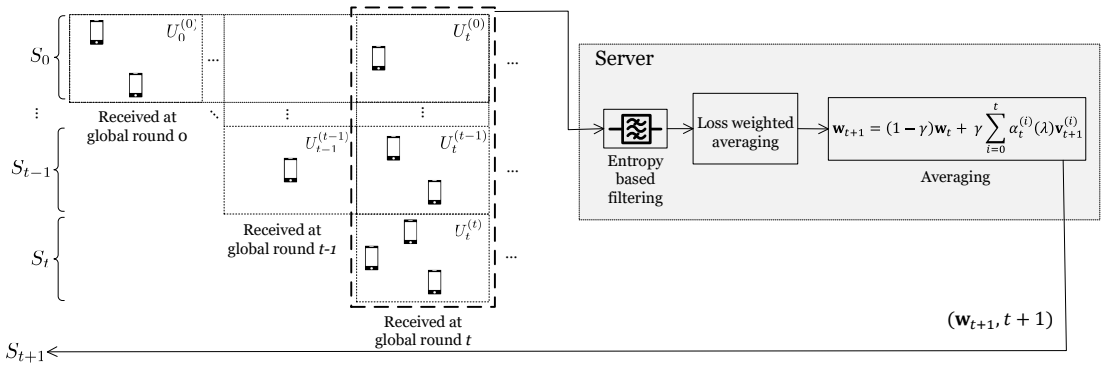



*Figure 1.* Overall procedure of Sageflow. At global round $t$, each received model belongs to one of the $t+1$ sets: $U_t^{(0)}, U_t^{(1)}, ..., U_t^{(t)}$. After entropy-based filtering, the server performs loss-weighted averaging for the results that belong to $U_t^{(i)}(E_{th})$ to obtain $\mathbf{v}_{t+1}^{(i)}$. Then we aggregate $\{\mathbf{v}_{t+1}^{(i)}\}_{i=0}^{t}$ with $\mathbf{w}_t$ to obtain $\mathbf{w}_{t+1}$, and move on to the next round $t+1$.

to the server at global round $i$ for $i \geq t$. Then, we can write $S_t = \cup_{i=t}^{\infty} U_i^{(t)}$, where $U_i^{(t)} \cap U_j^{(t)} = \emptyset$ for $i \neq j$. Here, $U_\infty^{(t)}$ can be viewed as the devices that are selected at round $t$ but failed to successfully send their results back to the server. According to these notations, the devices whose training results arrive at the server during global round $t$ belong to one of the $t+1$ groups: $U_t^{(0)}, U_t^{(1)}, ..., U_t^{(t)}$. Note that the result sent from device $k \in U_t^{(i)}$ is the model after $E$ local updates starting from $\mathbf{w}_i$, and we denote this model by $\mathbf{w}_i(k)$. At each round $t$, we first perform FedAvg as

$$\mathbf{v}_{t+1}^{(i)} = \sum_{k \in U_t^{(i)}} \frac{m_k}{\sum_{k \in U_t^{(i)}} m_k} \mathbf{w}_i(k) \tag{2}$$

for all $i = 0, 1, ..., t$. Here, $\mathbf{v}_{t+1}^{(i)}$ can be viewed as a group representative model, which is the aggregated result of locally updated models (starting from $\mathbf{w}_i$) received at round $t$ with staleness $t-i+1$. Then from the representative models of all $t$ groups, $\mathbf{v}_{t+1}^{(0)}$, $\mathbf{v}_{t+1}^{(1)}$,..., $\mathbf{v}_{t+1}^{(t)}$, we take a weighted averaging according to different staleness: $\sum_{i=0}^{t} \alpha_t^{(i)}(\lambda)\mathbf{v}_{t+1}^{(i)}$. Here, $\alpha_t^{(i)}(\lambda) \propto \frac{\sum_{k \in U_t^{(i)}} m_k}{(t-i+1)^\lambda}$ is a normalized coefficient that is inversely proportional to $(t-i+1)^\lambda$, for a given staleness exponent $\lambda \geq 0$. Hence, we have a larger weight for $\mathbf{v}_{t+1}^{(i)}$ with a smaller $t-i+1$ (staleness). This is to give more weights to more recent results having smaller staleness. Based on the weighted sum $\sum_{i=0}^{t} \alpha_t^{(i)}(\lambda)\mathbf{v}_{t+1}^{(i)}$, we finally obtain $\mathbf{w}_{t+1}$ as

$$\mathbf{w}_{t+1} = (1-\gamma)\mathbf{w}_t + \gamma \sum_{i=0}^{t} \alpha_t^{(i)}(\lambda)\mathbf{v}_{t+1}^{(i)}, \tag{3}$$

where $\gamma$ is the time-average coefficient. Now we move on to the next round $t+1$, where the server selects $S_{t+1}$ and sends $(\mathbf{w}_{t+1}, t+1)$ to these devices. Here, if the server knows the set of active devices (which are still performing computation), $S_{t+1}$ can be constructed to be disjoint with the active devices. If not, the server randomly chooses $S_{t+1}$ among all devices and the selected active devices can ignore the current request of the server. The left-hand side of Fig. 1 describes our staleness-aware grouping method.

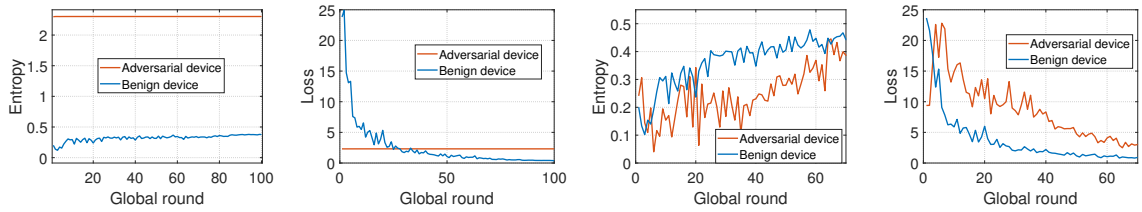
### 2.2. Entropy-based Filtering and Loss-Weighted Averaging against Adversaries

Now we propose a solution against adversaries which not only shows better performance with attacks but also combines well with our staleness-aware grouping scheme compared to existing aggregation methods to handle adversaries. We provide the following two solutions which can protect the system in a highly complementary fashion against various attacks using only a small amount of public data.

**1) Entropy-based filtering.** Let $n_{pub}$ be the number of public data samples in the server. We also let $x_{pub,j}$ be the $j$-th sample in the server. When the server receives the locally updated models from the devices, it measures the entropy of each device $k$ by utilizing the public data as

$$E(k) = \frac{1}{n_{pub}} \sum_{j=1}^{n_{pub}} E_{x_{pub,j}}(k), \tag{4}$$

where $E_{x_{pub,j}}(k)$ is the shannon entropy of the model of the $k$-th device on the sample $x_{pub,j}$ written as $E_{x_{pub,j}}(k) = -\sum_{q=1}^{Q} P_{x_{pub,j}}^{(q)}(k) \log P_{x_{pub,j}}^{(q)}(k)$. Here, $Q$ is the number of classes of the dataset and $P_{x_{pub,j}}^{(q)}(k)$ is the probability of prediction for the $q$-th class on a sample $x_{pub,j}$, using the model of the $k$-th device. As can be seen from Fig. 2(a) with FMNIST dataset, under specific model-update poisoning attacks (reverse sign attack with scale 0.1 in this case), the models compromised by adversaries tend to predict randomly for all classes and thus have high entropies compared to the models of benign devices. Based on this observation, we let the server to filter out the models that have entropies greater than some threshold value $E_{th}$. We note that the inflicted models cannot be filtered out based on the loss values in the setting of model update poisoning in Fig. 2(b), which confirms the importance of the role of entropy. We




(a) Model poison, entropy (b) Model poison, loss (c) Data poison, entropy (d) Data poison, loss

*Figure 2.* **Model update poisoning ((a), (b)):** We can filter out the models of adversaries via entropy, but not via loss. **Data poisoning ((c), (d)):** We can reduce the impact of adversaries via loss, but not via entropy.

also note that our entropy-based filtering is robust against attacks even with a large portion of adversaries, since it just filters out the results whose entropy is greater than $E_{th}$. This is a significant advantage compared to current aggregation methods against adversaries, whose performance are substantially degraded with high attack ratio.

**2) Loss-weighted averaging.** The server also measures the loss of each model using the public data. Based on the loss values, the server aggregates the models as follows:

$$\mathbf{w}_{t+1} = \sum_{k \in S_t} \beta_t^{(k)}(\delta)\mathbf{w}_t(k), \text{ where} \tag{5}$$

$$\beta_t^{(k)}(\delta) \propto \frac{m_k}{\{F_{pub}(\mathbf{w}_t(k))\}^{\delta}} \text{ and } \sum_{k \in S_t} \beta_t^{(k)}(\delta) = 1. \tag{6}$$

Here, $\mathbf{w}_t(k)$ is the locally updated model of the $k$-th device at global round $t$. $F_{pub}(\mathbf{w}_t(k))$ is defined as the averaged loss of $\mathbf{w}_t(k)$ computed on public data at the server, i.e., $F_{pub}(\mathbf{w}_t(k)) = \frac{1}{n_{pub}} \sum_{j=1}^{n_{pub}} \ell(\mathbf{w}_t(k); x_{pub,j})$. Finally, $\delta(\geq 0)$ in $\{F_{pub}(\cdot)\}^{\delta}$ is a loss exponent related to the impact of the loss measured with public data. We note that a setting $\delta = 0$ in (5) reduces our loss-weighted averaging to FedAvg. Under the data poisoning or model replacement backdoor attacks (or scaled backdoor attack) in (Bagdasaryan et al., 2018), the models of adversaries have relatively larger losses compared to others. Fig. 2(d) shows an example with FMNIST dataset under data poisoning attack. By the definition of $\beta_t^{(k)}(\delta)$, the data-poisoned model would be given a small weight and has a less impact on the next global update. By replacing FedAvg with the above loss-weighted averaging, we are able to build a system that is highly robust against data poisoning and scaled backdoor attacks. As can be seen from Fig. 2(c), the impact of data poisoning cannot be reduced via entropy measures. This indicates that the loss measure has its own unique role, along with the entropy.

Entropy-based filtering and loss-weighted averaging can be easily combined, and work complementarily to tackle model/data poisoning and scaled backdoor attacks.

### 2.3. Sageflow

Finally we put together Sageflow, which can handle both stragglers and adversaries at the same time by applying entropy-based filtering and loss-weighted averaging in each grouping stage of our straggler-mitigating staleness-aware aggregation. The details of overall Sageflow operation are described in Algorithm 1 and Fig. 1.

We stress that Sageflow performs well with only a very small amount of public data at the server, 2% of the entire dataset, which is comparable to the amount of local data at a single device. The computational complexity of Sageflow depends on the number of received models at each global round, and the running time for computing the entropy/loss of each model. Assuming that the complexity of computing entropy or loss is linear to the number of model parameters as in (Xie et al., 2019b), Sageflow has larger complexity than RFA by the factor $n_{pub}$. This small additional computation of Sageflow compared to RFA is the cost for considerably better robustness against adversaries.

## 3. Convergence Analysis

In this section, we provide insights into the convergence of Sageflow with the following standard assumptions in FL (Li et al., 2019a; Xie et al., 2019a).

**Assumption 1** *The global loss fuction $F$ defined in (1) is $\mu$-strongly convex, i.e., $F(x) \leq F(y) + \nabla F(x)^T(x-y) - \frac{\mu}{2}\|x-y\|^2$ for all $x$, $y$. Moreover, $F$ is $L$-smooth, i.e., $F(x) \geq F(y) + \nabla F(x)^T(x-y) - \frac{L}{2}\|x-y\|^2$ for all $x$, $y$.*

**Assumption 2** *Let $\mathbf{w}_t^i(k)$ be the model of the $k$-th benign device after $i$ local updates starting from global round $t$. Let $\xi_t^i(k)$ be a set of data samples that are randomly selected from the device $k$ for $(i+1)$-th local update. Then, $\mathbb{E}\|\nabla F_k(\mathbf{w}_t^i(k), \xi_t^i(k)) - \nabla F(\mathbf{w}_t^i(k))\|^2 \leq \rho_1$ holds for all $t$ and $k = 1, \ldots, N$ and $i = 1, \ldots, E$.*

Let $B_t^{(i)}$ and $M_t^{(i)}$ be the set for benign and adversarial devices of $U_t^{(i)}$ respectively, satisfying $U_t^{(i)} = B_t^{(i)} \cup M_t^{(i)}$ and $B_t^{(i)} \cap M_t^{(i)} = \emptyset$. Similarly, define $B_t^{(i)}(E_{th})$ and $M_t^{(i)}(E_{th})$ as the sets obtained after entropy-based filtering is applied to $B_t^{(i)}$ and $M_t^{(i)}$. Now we have the following definition which describes the effect of the adversaries.

**Definition 1** *We define $\Omega_t^{(i)}(E_{th}, \delta)$ as*

$$\Omega_t^{(i)}(E_{th}, \delta) = \sum_{k \in M_t^{(i)}(E_{th})} \beta_i^{(k)}(\delta)[F(\mathbf{w}_t(k)) - F(\mathbf{w}^*)]. \tag{7}$$

**Algorithm 1** Proposed Sageflow Algorithm

**Input:** Initialized model $\mathbf{w}_0$, **Output:** Final global model $\mathbf{w}_T$

**Process at the Server**

**for** each global round $t = 0, 1, ..., T-1$ **do**
  Choose $S_t$ and send the current model and the global round $(\mathbf{w}_t, t)$ to the devices
  Wait for $T_d$ and then:
  **for** $i = 0, 1, ..., t$ **do**
    $U_t^{(i)}(E_{th}) = \{k \in U_t^{(i)} | E(k) < E_{th}\}$ // Entropy-based filtering in each group
    $\mathbf{v}_{t+1}^{(i)} = \sum_{k \in U_t^{(i)}(E_{th})} \beta_t^{(k)}(\delta) \mathbf{w}_i(k)$ // Loss-weighted averaging in each group (with same staleness)
  **end for**
  $\mathbf{w}_{t+1} = (1-\gamma)\mathbf{w}_t + \gamma \sum_{i=0}^{t} \alpha_t^{(i)}(\lambda) \mathbf{v}_{t+1}^{(i)}$ // Averaging of representative models (with different staleness)
**end for**

**Process at the Device**: Device $k$ receives $(\mathbf{w}_t, t)$ from the server and performs local updates to obtain $\mathbf{w}_t(k)$. Then each benign device $k$ sends $(\mathbf{w}_t(k), t)$ to the server, while a malicious adversary sends a poisoned model depending on the type of attack.

Now we state the following theorem which provides the convergence bound of Sageflow.

**Theorem 1** *Suppose Assumptions 1, 2 hold and the learning rate $\eta$ is set to be less than $\frac{1}{L}$. Then Sageflow satisfies*

$$\mathbb{E}[F(\mathbf{w}_T) - F(\mathbf{w}^*)] \leq \nu^T [F(\mathbf{w}_0) - F(\mathbf{w}^*)] + (1-\nu^T) Z(\lambda, E_{th}, \delta) \quad (8)$$

*where* $\nu = 1 - \gamma + \gamma(1-\eta\mu)^E$,

$$Z(\lambda, E_{th}, \delta) = \frac{\rho_1 + 2\mu G_{max}(\lambda) + 2\mu \Omega_{max}(E_{th}, \delta)}{2\eta\mu^2}, \quad (9)$$

$G_{max}(\lambda) = \max_{1 \leq t \leq T} \sum_{i=0}^{t-1} \alpha_t^{(i)}(\lambda) e_t^{(i)}$, $\Omega_{max}(E_{th}, \delta) = \max_{t, i \leq t} \Omega_t^{(i)}(E_{th}, \delta)$, $e_t^{(i)} = F(\mathbf{w}_i) - F(\mathbf{w}_t)$.

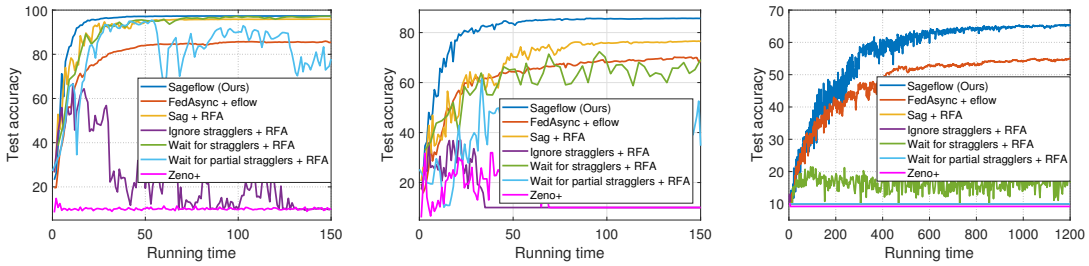
In (8) above, we see a trade-off between $\nu^T$, which determines the convergence speed, and $(1-\nu^T)Z$, which represents an error. If we increase $\gamma$, we have a higher convergence speed (i.e., a small $\nu^T$) but a larger error term. Our scheme allows a separate control on the error term $Z(\lambda, E_{th}, \delta)$ through staleness exponent $\lambda$, entropy-filtering threshold $E_{th}$ and the loss-weighted exponent $\delta$. Here, in $Z(\lambda, E_{th}, \delta)$ of (9), $G_{max}(\lambda)$ is the error term caused by stragglers controlled by $\lambda$, and $\Omega_{max}(E_{th}, \delta)$ is the errror caused by adversaries controlled by $E_{th}$ and $\delta$. First, to gain insights on $G_{max}(\lambda)$, assume that the loss of the global model decreases as global round increases, i.e., $F(\mathbf{w}_i) < F(\mathbf{w}_j)$ for $i > j$. Then, we have $e_t^{(0)} > e_t^{(1)} > ..., > e_t^{(t-1)} > 0$. In order to reduce $\sum_{i=0}^{t-1} \alpha_t^{(i)}(\lambda) e_t^{(i)}$, we have to increase the staleness exponent $\lambda$ to increase $\alpha_t^{(i)}(\lambda)$ for a large $i$ (weight for the group with small staleness) while reducing $\alpha_t^{(i)}(\lambda)$ for a small $i$ (weight for the group with large staleness). By choosing an appropriate $\lambda$, we can control the error term $G_{max}$ while taking advantage of the results sent from stragglers. As for the adversary-induced error $\Omega_{max}(E_{th}, \delta)$, by imposing a threshold $E_{th}$, we can reduce the portion of adversaries (with high entropies) in each group $U_t^{(i)}(E_{th})$, which in turn reduces $\Omega_t^i$ according to (7). Likewise, by introducing $\delta$, we can reduce the loss-weights $\beta_i^{(k)}(\delta)$ (defined in (6)) of the adversaries with large losses, which again reduces $\Omega_t^{(i)}$ according to (7). In the next section, we show via experiments that Sageflow in fact successfully combats both stragglers and adversaries simultaneously and achieves fast convergence with a small error term.

## 4. Experiments

We validate Sageflow on MNIST (LeCun et al., 1998), FMNIST (Xiao et al., 2017) and CIFAR10 (Krizhevsky et al., 2009). A simple convolutional neural network (CNN) with 2 convolutional layers and 2 fully connected layers is utilized for MNIST, while CNN with 2 convolutional layers and 1 fully connected layer is used for FMNIST. When training with CIFAR10, we utilized VGG-11.

We consider $N = 100$ devices each having the same number of data samples. We randomly assigned two classes to each device as in (McMahan et al., 2017) to create non-IID situations. We ignored the batch normalization layers when training VGG-11 with CIFAR10. At each global round, we randomly selected a fraction $C$ of devices in the system to participate. The portion of adversaries at each global round is $r$. For the proposed Sageflow method, we sample $2\%$ of the entire training data uniformly at random to be the public data and performed FL with the remaining $98\%$ of the train set. The number of local epochs at each device is set to 5.

**Comparisons schemes.** Under the existence of both stragglers and adversaries, we compare our Sageflow with various combinations of straggler/adversary mitigating schemes. Targeting stragglers, we consider the following methods. First is the *wait for stragglers* approach where FedAvg is applied after waiting for all the devices at each global round. The second scheme is the *ignore stragglers* approach where FedAvg is applied after waiting for a certain timeout threshold and ignore the results sent from slow devices. The third scheme is the *wait for a percantage of stragglers* where




(a) MNIST (b) FMNIST (c) CIFAR10

*Figure 3.* Model-update poisoning attack: Sageflow outperforms various combinations aiming to handle stragglers/adversaries.

*Table 1.* Performance at a specific time in Fig. 3.

| | Model update poisoning (Test accuracy) | | | Data poisoning (Test accuracy) | | | Scaled backdoor attack (Attack success rate) | | |
|---|---|---|---|---|---|---|---|---|---|
| **Methods\Datasets** | MNIST | FMNIST | CIFAR10 | MNIST | FMNIST | CIFAR10 | MNIST | FMNIST | CIFAR10 |
| Zeno+ | 9.96% | 9.95% | 9.21% | 12.27% | 10.00% | 10.00% | - | - | - |
| Ignore stragglers + RFA | 10.04% | 10.00% | 10.00% | 97.38% | 80.78% | 64.51% | - | - | - |
| Wait for partial stragglers + RFA | 78.29% | 35.29% | 10.00% | 97.59% | 70.15% | 62.13% | 32.16% | 98.49% | 82.26% |
| Wait for stragglers + RFA | 96.20% | 70.01% | 17.17% | 96.60% | 80.01% | 58.51% | 3.17% | 42.79% | 81.74% |
| Sag + RFA | 95.78% | 73.64% | 10.00% | **97.60%** | 83.55% | 64.18% | 99.97% | 99.97% | 90.79% |
| FedAsync + eflow | 84.7% | 66.38% | 53.03% | 83.47% | 48.80% | 51.28% | 99.99% | 99.86% | 91.34% |
| **Sageflow (Ours)** | **97.27%** | **85.23%** | **63.76%** | 97.38% | **85.01%** | **64.87%** | **0.21%** | **3.79%** | **5.56%** |

FedAvg is applied after waiting for a specific portion of the selected devices in each global round. We consider a scheme that waits for 50% of selected devices. Finally, we consider FedAsync (Xie et al., 2019a) where the global model is updated every time the result of each device arrives. Targeting adversaries, we consider both RFA (Pillutla et al., 2019) and an asynchronous version of Zeno+ (as in (Xie et al., 2019c)) which utilizes public data at the server: the server first subtracts each survived model (after filtering) from the previous global model to obtain the model difference. Then, the global model is updated asynchronously based on each model difference. For a fair comparison, we let $2\%$ of the train set to be the public data and the remaining $98\%$ to be distributed at the devices, as in our Sageflow.

**Setup.** The global aggregation at the server is performed with every $T_d = 1$ periodically for Sageflow, ignore stragglers and FedAsync. To model stragglers, each device can have delay of 0, 1, 2 global rounds which is determined independently and uniformly random. For model update poisoning, each adversary sends $-0.1\mathbf{w}$ to the server, instead of sending the true model $\mathbf{w}$. For data poisoning attack, we conduct *label-flipping* (Biggio et al., 2012), where each label $i$ is flipped to label $i + 1$. For the backdoor, we use the *model replacement* method (scaled backdoor attack) (Bagdasaryan et al., 2018) in which adversaries transmit the scaled version of the corrupted model to replace the global model with a bad model. We conduct *pixel-pattern backdoor attack* (Gu et al., 2017) in which the specific pixels are embedded in a fraction of images, where these images are classified as a targeted label. We applied backdoor attack in every global round after the 10-th round for MNIST and FMNIST, and after the 1000-th round for CIFAR10.

**Results.** Fig. 3 and Table 1 show the results under the existence of both stragglers and adversaries. We set $C = 0.2$, $r = 0.2$ for model/data poisoning and $C = 0.1$, $r = 0.1$ for the backdoor attack. In backdoor attack, we excluded the results for Zeno+ and RFA combined with *ignore stragglers*, since the models are not trained at all. We have the following observations. First, Zeno+ does not perform well since it does not take both the staleness and entropy into account. It can be also seen that the *wait for stragglers* scheme combined with RFA suffers from the straggler issue. Our next observation is that the RFA combined with *ignore stragglers* method exhibits poor performance. The reason is that the attack ratio could often be very high (larger than $r$) for this deadline-based scheme, which degrades the performance of RFA. Due to the same issue, RFA combined with our staleness-aware grouping (Sag) has performance degradation. FedAsync does not perform well when combined with entropy-based filtering and loss-weighted averaging (eflow), since the model update is conducted one-by-one in the order of arrivals. Due to the same issue, FedAsync cannot be combined with RFA. Overall, the proposed Sageflow performs the best, confirming significant advantages of our scheme under the existence of both stragglers and adversaries.

## 5. Conclusion

We proposed Sageflow, a robust FL scheme handling both stragglers and adversaries simultaneously. The staleness-aware grouping allows the server to effectively utilize the results sent from stragglers. The grouping strategy also integrates naturally with defenses against adversaries. In each grouping stage of our straggler-mitigating idea, entropy-based filtering and loss-weighted averaging function in a complementary fashion to protect the system against a wide variety of adversarial attacks. Theoretical convergence anal-

ysis provides key insights into why Sageflow works well. Experimental results show that Sageflow enables robust FL in practical scenarios with both stragglers and adversaries.

## References


Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., and Shmatikov, V. How to backdoor federated learning. *arXiv preprint arXiv:1807.00459*, 2018.

Biggio, B., Nelson, B., and Laskov, P. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*, 2012.

Blanchard, P., Guerraoui, R., Stainer, J., et al. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems*, pp. 119–129, 2017.

Chen, X., Liu, C., Li, B., Lu, K., and Song, D. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017a.

Chen, Y., Su, L., and Xu, J. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(2):44, 2017b.

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

Konečnỳ, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

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

Li, Q., He, B., and Song, D. Model-agnostic round-optimal federated learning via knowledge transfer. *arXiv preprint arXiv:2010.01017*, 2020.

Li, X., Huang, K., Yang, W., Wang, S., and Zhang, Z. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*, 2019a.

Li, Y., Yang, S., Ren, X., and Zhao, C. Asynchronous federated learning with differential privacy for edge intelligence. *arXiv preprint arXiv:1912.07902*, 2019b.

Liu, Y., Ma, S., Aafer, Y., Lee, W.-C., Zhai, J., Wang, W., and Zhang, X. Trojaning attack on neural networks. 2017.

Lu, X., Liao, Y., Lio, P., and Hui, P. Privacy-preserving asynchronous federated learning mechanism for edge network computing. *IEEE Access*, 8:48970–48981, 2020.

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

Pillutla, K., Kakade, S. M., and Harchaoui, Z. Robust aggregation for federated learning. *arXiv preprint arXiv:1912.13445*, 2019.

van Dijk, M., Nguyen, N. V., Nguyen, T. N., Nguyen, L. M., Tran-Dinh, Q., and Nguyen, P. H. Asynchronous federated learning with reduced number of rounds and with differential privacy from less aggregated gaussian noise. *arXiv preprint arXiv:2007.09208*, 2020.

Wu, W., He, L., Lin, W., Jarvis, S., et al. Safa: a semi-asynchronous protocol for fast federated learning with low overhead. *arXiv preprint arXiv:1910.01355*, 2019.

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

Xie, C., Koyejo, S., and Gupta, I. Asynchronous federated optimization. *arXiv preprint arXiv:1903.03934*, 2019a.

Xie, C., Koyejo, S., and Gupta, I. Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance. In *International Conference on Machine Learning*, pp. 6893–6901. PMLR, 2019b.

Xie, C., Koyejo, S., and Gupta, I. Zeno++: Robust fully asynchronous sgd. *arXiv preprint arXiv:1903.07020*, 2019c.

Yin, D., Chen, Y., Ramchandran, K., and Bartlett, P. Byzantine-robust distributed learning: Towards optimal statistical rates. *arXiv preprint arXiv:1803.01498*, 2018a.

Yin, D., Chen, Y., Ramchandran, K., and Bartlett, P. Defending against saddle point attack in byzantine-robust distributed learning. *arXiv preprint arXiv:1806.05358*, 2018b.

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.

# A. Proof of Theorem 1

## A.1. Additional Notations for Proof

Let $\mathbf{w}_t^j(k)$ be the model of the $k$-th benign device after $j$ local updates starting from global round $t$. At global round $t$, each device receives the current global model $\mathbf{w}_t$ and round index (time stamp) $t$ from the server, and sets its initial model to $\mathbf{w}_t$, i.e., $\mathbf{w}_t^0(k) \leftarrow \mathbf{w}_t$ for all $k = 1, \ldots, N$. Then each $k$-th benign device performs $E$ local updates of stochastic gradient descent (SGD) with learning rate $\eta$:

$$\mathbf{w}_t^j(k) \leftarrow \mathbf{w}_t^{j-1}(k) - \eta \nabla F_k(\mathbf{w}_t^{j-1}(k), \xi_t^{j-1}(k)) \; for \; j = 1, \ldots, E, \tag{10}$$

where $\xi_t^j(k)$ is a set of data samples that are randomly selected from the $k$-th device during the $j$-th local update at global round $t$. After $E$ local updates, the $k$-th benign device transmits $\mathbf{w}_t^E(k)$ to the server. However, in each round, the adversarial devices transmit poisoned model parameters.

Using these notations, the parameters defined in Section 2 can be rewritten as follows:

$$\mathbf{v}_{t+1}^{(i)} = \sum_{k \in U_t^{(i)}(E_{th})} \beta_i^{(k)}(\delta) \mathbf{w}_i^E(k) \;\; \text{where} \;\; \beta_i^{(k)}(\delta) \propto \frac{m_k}{\{F_{pub}(\mathbf{w}_i^E(k))\}^\delta} \;\; \text{and} \sum_{k \in U_t^{(i)}(E_{th})} \beta_i^{(k)}(\delta) = 1 \tag{11}$$

$$\mathbf{z}_{t+1} = \sum_{i=0}^{t} \alpha_t^{(i)}(\lambda) \mathbf{v}_{t+1}^{(i)} \quad \text{where} \;\; \alpha_t^{(i)}(\lambda) \propto \frac{\sum_{k \in U_t^{(i)}} m_k}{(t-i+1)^\lambda} \;\; \text{and} \;\; \sum_{i=0}^{t} \alpha_t^{(i)}(\lambda) = 1 \tag{12}$$

$$\mathbf{w}_{t+1} = (1-\gamma)\mathbf{w}_t + \gamma \mathbf{z}_{t+1} \tag{13}$$

We also define

$$\Gamma_{t-1}^{(i)} = \sum_{B_{t-1}^{(i)}(E_{th})} \beta_i^{(k)}(\delta) \tag{14}$$

where $0 \leq \Gamma_{t-1}^{(i)} \leq 1$.

## A.2. Key Lemma and Proof

We introduce the following key lemma for proving Theorem 1. A part of our proof is based on the convergence proof of FedAsync in (Xie et al., 2019a).

**Lemma 1** *Suppose Assumptions 1, 2 hold and the learning rate $\eta$ is set to be less than $\frac{1}{L}$. Consider the $k$-th benign device that received the current global model $\mathbf{w}_t$ from the server at global round $t$. After $E$ local updates, the following holds:*

$$\mathbb{E}[F(\mathbf{w}_t^E(k)) - F(\mathbf{w}^*) | \mathbf{w}_t^0(k)] \leq (1-\eta\mu)^E [F(\mathbf{w}_t^0(k)) - F(\mathbf{w}^*)] + \frac{E\rho_1\eta}{2}. \tag{15}$$

*Proof of Lemma 1.* First, consider one step of SGD in the $k$-th local device. For a given $\mathbf{w}_t^j(k)$, for all global round $t$ and for all local updates $j \in \{0, 1, \ldots, E-1\}$, we have

$$\begin{aligned}
&\mathbb{E}[F(\mathbf{w}_t^{j+1}(k)) - F(\mathbf{w}^*) | \mathbf{w}_t^j(k)] \\
&\leq F(\mathbf{w}_t^j(k)) - F(\mathbf{w}^*) - \eta \mathbb{E}[\nabla F(\mathbf{w}_t^j(k))^T \nabla F_k(\mathbf{w}_t^j(k), \xi_t^j(k)) | \mathbf{w}_t^j(k)] \\
&\quad + \frac{L\eta^2}{2} \mathbb{E}[\|\nabla F_k(\mathbf{w}_t^j(k), \xi_t^j)\|^2 | \mathbf{w}_t^j(k)] && \blacktriangleright \text{ SGD update and } L\text{-smoothness} \\
&\leq F(\mathbf{w}_t^j(k)) - F(\mathbf{w}^*) + \frac{\eta}{2} \mathbb{E}[\|\nabla F(\mathbf{w}_t^j(k)) - \nabla F_k(\mathbf{w}_t^j(k), \xi_t^j(k))\|^2 | \mathbf{w}_t^j(k)] \\
&\quad - \frac{\eta}{2} \|\nabla F(\mathbf{w}_t^j(k))\|^2 && \blacktriangleright \eta < \frac{1}{L} \\
&\leq F(\mathbf{w}_t^j(k)) - F(\mathbf{w}^*) - \frac{\eta}{2} \|\nabla F(\mathbf{w}_t^j(k))\|^2 + \frac{\eta\rho_1}{2} && \blacktriangleright \text{ Assumption 2} \\
&\leq (1-\eta\mu)[F(\mathbf{w}_t^j(k)) - F(\mathbf{w}^*)] + \frac{\eta\rho_1}{2} && \blacktriangleright \mu\text{-strongly convexity}
\end{aligned} \tag{16}$$

Applying above result to $E$ local updates in $k$-th local device, we have

$$
\begin{aligned}
&\mathbb{E}\left[F(\mathbf{w}_t^E(k)) - F(\mathbf{w}^*)|\mathbf{w}_t^0(k)\right] \\
&= \mathbb{E}[\ \mathbb{E}[F(\mathbf{w}_t^E(k)) - F(\mathbf{w}^*)|\mathbf{w}_t^{E-1}(k)]|\mathbf{w}_t^0(k)\ ] && \blacktriangleright \text{ Law of total expectation} \\
&\leq (1-\eta\mu)\mathbb{E}[[F(\mathbf{w}_t^{E-1}(k)) - F(\mathbf{w}^*)]|\mathbf{w}_t^0(k)] + \frac{\eta\rho_1}{2} && \blacktriangleright \text{ Inequality (16)} \\
&\qquad\qquad \vdots \\
&\leq (1-\eta\mu)^E[F(\mathbf{w}_t^0(k)) - F(\mathbf{w}^*)] + \frac{\eta\rho_1}{2}\sum_{j=1}^{E}(1-\eta\mu)^{j-1} \\
&= (1-\eta\mu)^E[F(\mathbf{w}_t^0(k)) - F(\mathbf{w}^*)] + \frac{\eta\rho_1}{2}\frac{1-(1-\eta\mu)^E}{\eta\mu} && \blacktriangleright \text{ From } \eta < \frac{1}{L} \leq \frac{1}{\mu},\ \eta\mu < 1 \\
&\leq (1-\eta\mu)^E[F(\mathbf{w}_t^0(k)) - F(\mathbf{w}^*)] + \frac{E\eta\rho_1}{2} && \blacktriangleright \text{ From } \eta\mu < 1,\ 1-(1-\eta\mu)^E \leq E\eta\mu
\end{aligned}
$$

### A.3. Proof of Theorem 1

Now utilizing Lemma 1, we provide the proof for Theorem 1. First, consider one round of global aggregation at the server. For a given $\mathbf{w}_{t-1}$, the server updates the global model according to equation (13). Then for all $t \in 1, \ldots, T$, we have

$$
\begin{aligned}
&\mathbb{E}[F(\mathbf{w}_t) - F(\mathbf{w}^*)|\mathbf{w}_{t-1}] \\
&\overset{(a)}{\leq} (1-\gamma)[F(\mathbf{w}_{t-1}) - F(\mathbf{w}^*)] + \gamma\mathbb{E}[F(\mathbf{z}_t) - F(\mathbf{w}^*)|\mathbf{w}_{t-1}] \\
&\overset{(b)}{\leq} (1-\gamma)[F(\mathbf{w}_{t-1}) - F(\mathbf{w}^*)] + \gamma\sum_{i=0}^{t-1}\alpha_{t-1}^{(i)}(\lambda)\mathbb{E}[F(\mathbf{v}_t^{(i)}) - F(\mathbf{w}^*)|\mathbf{w}_{t-1}] \\
&\overset{(c)}{\leq} (1-\gamma)[F(\mathbf{w}_{t-1}) - F(\mathbf{w}^*)] + \gamma\sum_{i=0}^{t-1}\alpha_{t-1}^{(i)}(\lambda)\sum_{k\in U_{t-1}^{(i)}(E_{th})}\beta_i^{(k)}(\delta)\mathbb{E}[F(\mathbf{w}_i^E(k)) - F(\mathbf{w}^*)|\mathbf{w}_{t-1}] \\
&= (1-\gamma)[F(\mathbf{w}_{t-1}) - F(\mathbf{w}^*)] + \gamma\sum_{i=0}^{t-1}\alpha_{t-1}^{(i)}(\lambda)\Big\{\sum_{k\in B_{t-1}^{(i)}(E_{th})}\beta_i^{(k)}(\delta)\mathbb{E}[F(\mathbf{w}_i^E(k)) - F(\mathbf{w}^*)|\mathbf{w}_{t-1}] \\
&\quad + \sum_{k\in M_{t-1}^{(i)}(E_{th})}\beta_i^{(k)}(\delta)\mathbb{E}[F(\mathbf{w}_i^E(k)) - F(\mathbf{w}^*)|\mathbf{w}_{t-1}]\Big\} \\
&\overset{(d)}{\leq} (1-\gamma+\gamma\alpha_{t-1}^{(t-1)}(\lambda)\Gamma_{t-1}^{(t-1)}(1-\eta\mu)^E)[F(\mathbf{w}_{t-1}) - F(\mathbf{w}^*)] + \frac{E\eta\rho_1\gamma}{2} \\
&\quad + \gamma(1-\eta\mu)^E\sum_{i=0}^{t-2}\alpha_{t-1}^{(i)}(\lambda)\sum_{k\in B_{t-1}^{(i)}(E_{th})}\beta_i^{(k)}(\delta)\left[F(\mathbf{w}_i^0(k)) - F(\mathbf{w}^*)\right] \\
&\quad + \gamma\sum_{i=0}^{t-1}\alpha_{t-1}^{(i)}(\lambda)\sum_{k\in M_{t-1}^{(i)}(E_{th})}\beta_i^{(k)}(\delta)\mathbb{E}[F(\mathbf{w}_i^E(k)) - F(\mathbf{w}^*)|\mathbf{w}_{t-1}] \\
&\overset{(e)}{\leq} (1-\gamma+\gamma\alpha_{t-1}^{(t-1)}(\lambda)\Gamma_{t-1}^{(t-1)}(1-\eta\mu)^E)[F(\mathbf{w}_{t-1}) - F(\mathbf{w}^*)] + \frac{E\eta\rho_1\gamma}{2} \\
&\quad + \gamma(1-\eta\mu)^E\sum_{i=0}^{t-2}\alpha_{t-1}^{(i)}(\lambda)\sum_{k\in B_{t-1}^{(i)}(E_{th})}\beta_i^{(k)}(\delta)\left[\underbrace{F(\mathbf{w}_i) - F(\mathbf{w}^*)}_{F(\mathbf{w}_i)-F(\mathbf{w}_{t-1})+F(\mathbf{w}_{t-1})-F(\mathbf{w}^*)}\right] \\
&\quad + \gamma\Omega_{max}(E_{th},\delta) \\
&= (1-\gamma+\gamma\sum_{i=0}^{t-1}\alpha_{t-1}^{(i)}(\lambda)\Gamma_{t-1}^{(i)}(1-\eta\mu)^E)[F(\mathbf{w}_{t-1}) - F(\mathbf{w}^*)] + \frac{E\eta\rho_1\gamma}{2} \\
&\quad + \gamma(1-\eta\mu)^E\sum_{i=0}^{t-2}\alpha_{t-1}^{(i)}(\lambda)\sum_{k\in B_{t-1}^{(i)}(E_{th})}\beta_i^{(k)}(\delta)\left[F(\mathbf{w}_i) - F(\mathbf{w}_{t-1})\right] + \gamma\Omega_{max}(E_{th},\delta) \\
&\overset{(f)}{\leq} (1-\gamma+\gamma(1-\eta\mu)^E)[F(\mathbf{w}_{t-1}) - F(\mathbf{w}^*)] + \frac{E\eta\rho_1\gamma}{2} \\
&\quad + \gamma G_{t-1}(\lambda) + \gamma\Omega_{max}(E_{th},\delta)
\end{aligned}
\tag{17}
$$

where $G_t(\lambda) := \sum_{i=0}^{t-1}\alpha_t^{(i)}(\lambda)e_t^{(i)}$ and we define $G_0(\lambda) = 0$. $(a)$, $(b)$, $(c)$ come from convexity, $(d)$ follows Lemma 1, $(e)$ comes from $\Omega_{max} = \max_{0\leq i\leq t, 0\leq t\leq T}\Omega_t^{(i)}$. $(f)$ comes from the fact that $\eta\mu < 1$ and $0 \leq \alpha_t^{(i)}(\lambda) \leq 1$ and $0 \leq \Gamma_t^{(i)} \leq 1$ for all $i$, $t$ and $\sum_{i=0}^{t}\alpha_t^{(i)}(\lambda) = 1$ for all $t$.

Applying the above result to $T$ global aggregations in the server, we have

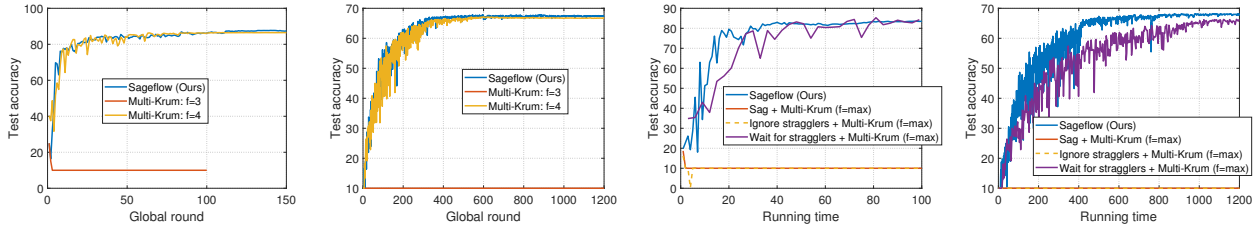
$$
\begin{aligned}
&\mathbb{E}[F(\mathbf{w}_T) - F(\mathbf{w}^*)|\mathbf{w}_0] \\
&\overset{(a)}{=} \mathbb{E}\left[\mathbb{E}[F(\mathbf{w}_T) - F(\mathbf{w}^*)|\mathbf{w}_{T-1}]|\mathbf{w}_0\right] \\
&\overset{(b)}{\leq} \mathbb{E}\left[(1-\gamma+\gamma(1-\eta\mu)^E)[F(\mathbf{w}_{T-1}) - F(\mathbf{w}^*)]|\mathbf{w}_0\right] + \frac{\gamma(E\eta\rho_1 + 2G_{T-1}(\lambda) + 2\Omega_{max}(E_{th},\delta))}{2} \\
&\overset{(c)}{\leq} (1-\gamma+\gamma(1-\eta\mu)^E)^T[F(\mathbf{w}_0) - F(\mathbf{w}^*)] + \frac{\gamma(E\eta\rho_1 + 2G_{T-1}(\lambda) + 2\Omega_{max}(E_{th},\delta))}{2} \\
&+ \sum_{\tau=1}^{T-1} \frac{\gamma(E\eta\rho_1 + 2G_{T-1-\tau}(\lambda) + 2\Omega_{max}(E_{th},\delta))}{2}(1-\gamma+\gamma(1-\eta\mu)^E)^\tau \\
&\overset{(d)}{\leq} (1-\gamma+\gamma(1-\eta\mu)^E)^T[F(\mathbf{w}_0) - F(\mathbf{w}^*)] \\
&\quad + \left[1 - \{1-\gamma+\gamma(1-\eta\mu)^E\}^T\right] \frac{E\eta\rho_1 + 2G_{max}(\lambda) + 2\Omega_{max}(E_{th},\delta)}{2(1-(1-\eta\mu)^E)} \\
&\overset{(e)}{\leq} (1-\gamma+\gamma(1-\eta\mu)^E)^T[F(\mathbf{w}_0) - F(\mathbf{w}^*)] \\
&\quad + \left[1 - \{1-\gamma+\gamma(1-\eta\mu)^E\}^T\right] \frac{\rho_1 + 2\mu G_{max}(\lambda) + 2\mu\Omega_{max}(E_{th},\delta)}{2\eta\mu^2} \\
&= \nu^T[F(\mathbf{w}_0) - F(\mathbf{w}^*)] + (1-\nu^T)Z(\lambda, E_{th}, \delta)
\end{aligned}
$$

which completes the proof. Here, $(a)$ comes from the *Law of total expectation*, $(b)$, $(c)$ are due to inequality (17). $(d)$ is obtained from the definition of $G_{max}(\lambda) := \max_{1\leq t\leq T} \sum_{i=0}^{t-1} \alpha_t^{(i)}(\lambda) e_t^{(i)}$. In addition, $(e)$ is from $\eta\mu \leq 1$.

## B. Additional experimental results

### B.1. Performance comparison with Multi-Krum

While we compared Sageflow with RFA in our main manuscript, here we compare our scheme with *Multi-Krum* (Blanchard et al., 2017) which is a Byzantine-resilient aggregation method targeting conventional distributed learning setup with IID data across nodes. In Multi-Krum, among $N$ workers in the system, the server tolerates $f$ Byzantine workers under the assumption of $2f + 2 < N$. After filtering $f$ worker nodes based on squared-distances, the server chooses $M$ workers among $N - f$ remaining workers with the best scores and aggregates them. We set $M = N - f$ for comparing our scheme with Multi-Krum.




(a) FMNIST, only adversaries (b) CIFAR-10, only adversaries (c) FMNIST, both stragglers/adversaries (d) CIFAR-10, both stragglers/adversaries

*Figure 4.* Performance comparison with Multi-Krum under model-update poisoning. With only adversaries, Multi-Krum performs well when an appropriate $f$ parameter value is chosen. However, the performance of Multi-Krum degrades significantly when stragglers exist (even when combined with straggler-mitigating schemes). This is because the attack ratio can become very high when combined with staleness-aware grouping or the ignoring stragglers scheme; the number of adversaries exceeds $f$, significantly degrading the performance of Multi-Krum. When Multi-Krum is combined with the wait for stragglers scheme, the performance is not degraded by adversaries but by waiting for slow devices.

Fig. 4 compares Sageflow with Multi-Krum under model-update poisoning with scale 10. The stragglers are modeled with

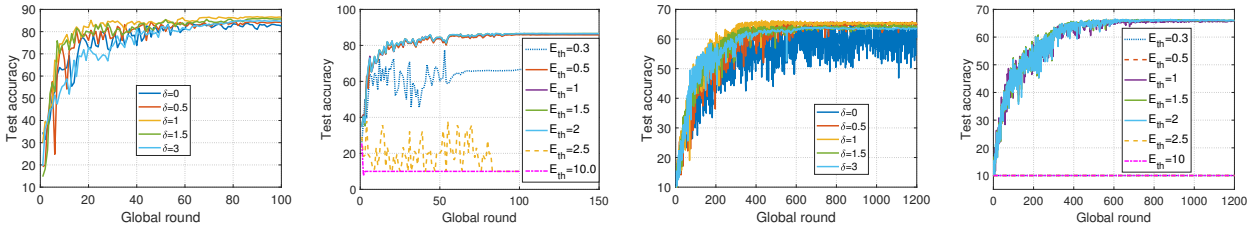
delay 0, 1, 2. We first observe Figs. 4(a) and 4(b) which show the results with only adversaries. It can be seen that if the number of adversaries exceed $f$, the performance of Multi-Krum drops dramatically. Compared to Multi-Krum, the proposed Sageflow method can filter out the poisoned devices and then take the weighted sum of the survived results even when the portion of adversaries is high. Figs. 4(c) and 4(d) show the results under the existence of both stragglers and adversaries, under the model-update poisoning attack. We let $C = 0.2$ and $r = 0.2$, and the parameter $f$ of Multi-Krum is set to the maximum value satisfying $2f + 2 < N$, where $N$ depends on the number of received results for both staleness-aware grouping (Sag) and *ignore stragglers* approaches. However, even when we set $f$ to the maximum value, the number of adversaries can still exceed $f$, which degrades the performance of Multi-Krum combined with staleness-aware grouping (Sag) or the *ignore stragglers* approach. Obviously, Multi-Krum can be combined with the *wait for stragglers* strategy by setting $f$ large enough. However, this scheme still suffers from the effect of stragglers, which significantly slows down the overall training process.

### B.2. Experimental results with varying hyperparameters

To observe the impact of hyperparameter setting, we performed additional experiments with various $\delta$ and $E_{th}$ values, the key hyperparameters of Sageflow. The results are shown in Fig. 5 with only adversaries. We performed the data poisoning attack for varying $\delta$ and the model-update poisoning attack with scale 0.1 for varying $E_{th}$. We set $C = 0.2$ and $r = 0.2$.

First, the results under data poisoning show that the performance of Sageflow is not sensitive to $\delta$ if they are chosen in the appropriate range of $[1, 2]$. For the model-update poisoning attack, if we use a very small $E_{th}$ like $0.3$, the performance is poor because a large number of devices get filtered out. If we use a large $E_{th}$, the performance is also very poor since the scheme cannot filter out the adversaries. However, similar to the behavior of hyperparameter $\delta$, we can confirm that our scheme performs well regardless of dataset if $E_{th}$ is chosen in an appropriate range of $[1, 2]$.

To summarize, our scheme still performs well (better than RFA), even with coarsely chosen hyperparameter values regardless of the dataset.




(a) FMNIST, data poisoning (b) FMNIST, model update poisoning (c) CIFAR-10, data poisoning (d) CIFAR-10, model update poisoning

*Figure 5.* Impact of varying hyperparameter values under model-update poisoning and data poisoning attacks. The performance of Sageflow is not highly sensitive to the exact settings of loss exponent $\delta$ and entropy threshold $E_{th}$, as long as they are chosen in a reasonable range.