Robust Combination of Distributed Gradients Under Adversarial Perturbations

Kwang In Kim
UNIST

Abstract

We consider distributed (gradient descent-based) learning scenarios where the server combines the gradients of learning objectives gathered from local clients. As individual data collection and learning environments can vary, some clients could transfer erroneous gradients e.g. due to adversarial data or gradient perturbations. Further, for data privacy and security, the identities of such affected clients are often unknown to the server. In such cases, naïvely aggregating the resulting gradients can mislead the learning process. We propose a new server-side learning algorithm that robustly combines gradients. Our algorithm embeds the local gradients into the manifold of normalized gradients and refines their combinations via simulating a diffusion process therein. The resulting algorithm is instantiated as a computationally simple and efficient weighted gradient averaging algorithm. In the experiments with five classification and three regression benchmark datasets, our algorithm demonstrated significant performance improvements over existing robust gradient combination algorithms as well as the baseline uniform gradient averaging algorithm.

1. Introduction

The success of deep learning relies on the abundance of training data and computational capability to process such data. When computational resources are limited at centralized servers, distributed learning can be employed, where data is distributed across multiple clients and part of the learning process is performed within each client. To enhance data privacy and security, and reduce the required communication cost, often in such environments, details of the localized data are hidden to the server and other clients, and the server aggregates only *encapsulated* information communicated from clients.

We consider distributed gradient descent-based optimization as an instance of such learning problems: The main objective is to minimize a differentiable energy function presented as the sum of the client energies defined based on the respective local data (e.g. the average of squared errors or cross-entry losses over clients' datasets) [23, 27, 41]. Then each client iteratively calculates and transfers the gradient of the local energy with respect to the learner parameters while the server maintains and updates these parameters using the aggregated local gradients.

The nature of individual clients and local data therein can vary significantly. For instance, a client could be a data center securing professionally annotated data while another client might be an edge device containing data labeled by inexperienced annotators. In this case, some clients could exhibit abnormal behavior e.g. due to incorrect labeling or adversarial attacks, and the corresponding local gradients can be unreliable. This can distract the gradient aggregation process at the server leading to suboptimal learning.

Robust distributed learning with such affected clients has only recently come to recognition and existing approaches are limited in that they require a separate training set at the server (to train a detector of affected clients) [34], information on the number of affected clients [32], or access to local training data [12] (see Sec. 2 for details).

We present a new approach that combines local gradients when some clients undergo adversarial perturbation. Our algorithm takes the naïve uniform average of gradients as a noisy observation of an underlying ground-truth combination, and iteratively improves this during the gradient descent steps of the learner parameters. Specifically, our algorithm represents the combined gradient as a convex combination of all local gradients. The corresponding combination weights are uniformly initialized. As the optimization proceeds, the genuine (unaffected) local gradients are automatically identified by adjusting the combination weights via simulating the diffusion of the combined gradient. Our approach does not require a separate training set at the server, and it offers the advantage of refining the gradient combination without having to know the number and identities of affected clients as well as the nature of perturbations therein.

In the experiments with five classification and three regression benchmark datasets, and two different scenarios of client perturbations, our approach demonstrated significant performance improvements over the standard uniform gradient averaging algorithm and existing robust gradient combination approaches.

2. Related work

Privacy-enhancing and communication-efficient distributed learning. Since the introduction of the federated averaging (*FedAvg*) algorithm [23], federated learning has been established as a standard problem and method for communication-efficient and cross-device distributed learning [17]. The main challenges in this scenario include minimizing the communication cost (e.g. via gradient compression [21] and quantization [19]), optimizing the topology of the underlying communication networks [43], data protection and encapsulation, personalization [7], and theoretical convergence guarantee [20, 35]. Recent work also focuses on robustness against non-independent and identically distributed (non-i.i.d.) data generation processes [24] and the extensions of *FedAvg* to e.g. multi-task learning [27] and meta-learning [18].

Robust distributed learning. In privacy and security-enhancing distributed learning, the server has only access to the gradients or updated learner parameters provided by the individual clients, and therefore, it is susceptible to corrupted updates [8, 39].

Learning approaches that are robust against such updates have only been recently introduced. Sun et al.'s gradient clipping approach assumes that the gradients generated from the affected clients tend to have large norms and it excludes the gradients whose norms are larger than a given threshold T in the combination [29]. This algorithm is tailored to attacks that focus on specific sub-tasks in multi-task environments, and it is not directly applicable to perturbations that we consider in this paper. Furthermore, in general adversarial perturbation scenarios, the threshold T might not be known in advance. Hu et al.'s gradient normalization approach builds on a similar idea [15]: Assuming that the affected gradients exhibit larger norms, they normalized the gradients to suppress their contributions. In our preliminary experiments (with random labeling and random gradient perturbations; see Sec. 4) we observed that the perturbed gradients do not necessarily exhibit larger norms than genuine gradients, and in such cases, gradient normalization or clipping are not directly applicable.

In Han and Zhang's robust federated learning algorithm, the clients take the role of (partially corrupted) teachers while the server acts as a student [12]. Each teacher then provides a few known genuine data instances, verified by domain experts, to guide the teaching process. This approach requires access to local data of individual clients and known identities of genuine data instances therein, and therefore its application is limited when the server is agnostic to the client internals.

The local gradients can be considered as abstract high-dimensional vectors and in this respect, combining perturbed local gradients can be seen as the problem of robustly estimating their mean (see Eq. 3). In general, when observed data are noisy, potentially including outliers, the central statistical tendency can be better captured by their medians as robust estimators than the means. Pillutla et al.'s gradient aggregation algorithm uses geometric median as another robust alternative to the mean [25]. In the experiments, we demonstrate that our approach outperforms the algorithms that replace the gradient means with their medians or geometric medians [25].

Turan et al.'s gradient combination algorithm provides an alternative approach to exploit the median estimates [32]: For each optimization step, the median of the local gradients are calculated. Then, the genuine gradients are selected as K-nearest neighbors of the median. While this algorithm demonstrated significant improvements over federated averaging, it requires determining K which might vary across the problems and be difficult to estimate. In the experiments, we show that this algorithm achieves robustness when the numbers of affected clients are limited and when K is known explicitly. However, when more clients are affected, the median estimates were distracted by the perturbed gradients and the performance degraded accordingly. Our algorithm demonstrated improvements over [32] even without having to tune K, especially when large numbers of clients are affected. Other robust mean alternatives can be found e.g. in gradient filtering via low-rank signal recovery [28] and trimmed mean [38, 42].

Some existing approaches build upon additional assumptions or trade data privacy and security with robustness: For example, Xie et al.'s Zeno algorithm detects affected clients based on the accuracy of each local model assessed based on labeled datasets at the server [40]. Alistarh et al.'s Byzantine stochastic gradient descent approach provides a theoretical guarantee on the near-optimality of the final solution, but it requires the convexity of training objectives [1]. Wan and Chen's robust federated learning algorithm detects the genuine gradients via training a separate learner that estimates the probability of each local gradient being genuine [34]. While this approach achieved a significant performance gain over uniform gradient averaging, its application domain is limited as it requires a separate training set at the server.

3. Robust gradient combination against adversarial perturbations

Problem formulation. Our goal is to learn a function f with domain $\mathcal{X}$ and range $\mathcal{Y}$. We will focus on deep neural network learners f parameterized by weight vectors $\mathbf{w}$ while our methods can be applied to other types of learners that can be trained using gradient descent techniques. For classification, we will use one-hot encoding where $\mathcal{Y} \subset \mathbb{R}^c$ with c being the number of classes. For regression, $\mathcal{Y} \subset \mathbb{R}$. In standard supervised learning, one is provided with a single training set $Z = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_N, \mathbf{y}_N)\} \subset \mathcal{X} \times \mathcal{Y}$ and the prediction function f is constructed by minimizing the sum of losses:

$$\mathcal{E}(\mathbf{w}; Z) = \sum_{j=1}^{N} l(f(\mathbf{x}_j), \mathbf{y}_j).$$

For classification, we will employ the standard cross-entropy loss: $l(\mathbf{a},\mathbf{b}) = -\sum_{j=1}^{c}[\mathbf{a}]_j \log([\mathbf{b}]_j)$ with $[\mathbf{a}]_j$ being the j-th element of the vector $\mathbf{a}$ while for regression, the squared error will be used: $l(\mathbf{a},\mathbf{b}) = (\mathbf{a}-\mathbf{b})^2$. Applying the standard gradient descent, one obtains the final learner parameter (weight vector) $\mathbf{w}^*$ by iteratively updating $\mathbf{w}$

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \eta \nabla_{\mathbf{w}} \mathcal{E}(\mathbf{w}; Z) \tag{1}$$

with a step size $\eta > 0$.

In distributed learning, the training set Z is distributed across K different clients $\{C_k\}_{k=1}^{K}$ where C_k is provided with a subset Z_k of Z, and it calculates the update of $\mathbf{w}$ based only on Z_k. For instance, McMahan et al.'s federated averaging (*FedAvg*) framework [23] replaces the centralized update in Eq. 1 with an aggregation of *local gradients*:[1]

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \eta \sum_{k=1}^{K} \mathbf{g}_k(t), \tag{2}$$

where $\mathbf{g}_k(t)$ is the gradient $\nabla_{\mathbf{w}} \mathcal{E}(\mathbf{w}(t); Z_k)$ calculated by the client C_k based on the learner parameter $\mathbf{w}(t)$. For stochastic gradient descent, only a small portion (called mini-batch) of Z_k contributes to the gradient evaluation $\nabla_{\mathbf{w}} \mathcal{E}$: All gradient combination algorithms considered in this work can be straightforwardly applied to this case. At iteration t, a server S distributes the previous learner parameter $\mathbf{w}(t)$, collects the resulting local gradients $\{\mathbf{g}_k\}_{k=1}^{K}$, and performs the parameter update in Eq. 2. To enhance security and privacy, the local datasets $\{Z_k\}$ are not shared with the server or other clients. Therefore, the only information that is communicated from C_k to S is the gradient $\mathbf{g}_k(t)$.

When each client provides a *genuine* gradient of $\mathcal{E}$, the *FedAvg* update $\mathbf{w}(t+1)$ in Eq. 2 is equivalent to the centralized update (Eq. 1) and they will lead to identical solutions. However, when some clients undergo adversarial perturbation and accordingly transfer erroneous gradients, the final solution of *FedAvg* process can be inferior to the centralized counterpart. The nature of the perturbations (e.g. the labels $\{\mathbf{y}_j\}_{j=1}^{|Z_k|}$ in Z_k are randomly changed or the k-th client gradient $\mathbf{g}_k(t)$ is replaced by a random vector; see Sec. 4) and the number and identities of the affected clients are not known to the server where the parameter update (Eq. 2) is performed.

Robust estimation of gradient update using diffusion on the gradient manifold. We employ a modified parameter update rule:

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \eta K \mathbf{q}(t), \tag{3}$$

$$\mathbf{q}(t) = \sum_{k=1}^{K} [\mathbf{a}(t)]_k \mathbf{g}_k(t), \tag{4}$$

where $\mathbf{a}(t)$ ($\|\mathbf{a}(t)\|_1 = 1$ and $[\mathbf{a}(t)]_k \geq 0$ for $1 \leq k \leq K$) regulates the contributions of the local gradients $\{\mathbf{g}_k(t)\}$. When $\mathbf{a}(t)$ is uniform (i.e. $[\mathbf{a}(t)]_k = 1/K$), Eq. 3 becomes the standard *FedAvg* (Eq. 2) while general non-uniform $\mathbf{a}(t)$ can instantiate different gradient combination strategies. Our algorithm automatically determines $\mathbf{a}(t)$ via simulating a diffusion process on the manifold M of normalized gradients. *Diffusion on data manifolds.* The diffusion of a function f on a manifold M induced by its Laplace-Beltrami operator Δ_M is described as a partial differential equation [16, 26]:

$$\frac{\partial h}{\partial t} = \Delta_M h. \tag{5}$$

When h is a noisy observation of an underlying ground-truth function $\overline{h}$

$$h(\mathbf{r}) = \overline{h}(\mathbf{r}) + \varepsilon, \tag{6}$$

with ε being Gaussian noise, it can be shown that the diffusion (Eq. 5) performs *denoising* of h depending on the curvature of M [14, 36]. In particular, when h is presented as a noisy embedding of M into an ambient Euclidean space $\mathbb{R}^m$, applying the diffusion in Eq. 5 tends to recover the underlying true embedding $\{\overline{h}(\mathbf{r}) : \mathbf{r} \in M\} \subset \mathbb{R}^m$ by *pushing along* M the noisy observations $\{h(\mathbf{r}) : \mathbf{r} \in M\} \subset \mathbb{R}^m$ towards them.

As shown shortly, we will take the local client gradients $\{\mathbf{g}_k\}$ and the combined gradient $\mathbf{q}$ (Eq. 4) as a sample from M and denoise $\mathbf{q}$ via simulating diffusion. For this, we first spatially discretize the original diffusion process: Suppose that we have a set of data instances $\overline{H} = \{\overline{\mathbf{h}}_1, \ldots, \overline{\mathbf{h}}_S\}$ sampled from a manifold M. The dataset $\overline{H}$ is not directly observed but instead presented as a noisy embedding $H = \{\mathbf{h}_1, \ldots, \mathbf{h}_S\}$ within $\mathbb{R}^m$. The normalized graph Laplacian $\mathbf{L}$ on H is defined as

$$\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A}, \text{ where} \tag{7}$$

$$[\mathbf{A}]_{ij} = \kappa(\mathbf{h}_i, \mathbf{h}_j; \sigma^2) := \exp\left(-\frac{\|\mathbf{h}_i - \mathbf{h}_j\|^2}{\sigma^2}\right), \tag{8}$$

for $i \neq j$ and $[\mathbf{A}]_{ij} = 0$ for $i = j$. The i,j-th kernel matrix entry $[\mathbf{A}]_{ij}$ measures the similarity of data instance $\mathbf{h}_i$ and $\mathbf{h}_j$ controlled by the parameter σ^2. $\mathbf{D}$ is a diagonal matrix of the column-sums of $\mathbf{A}$: $[\mathbf{D}]_{ii} = \sum_j [\mathbf{A}]_{ij}$, which makes the corresponding diffusion process (presented below) probabilistically normalized. With graph Laplacian $\mathbf{L}$ as an approximation of $-\Delta_M$, a spatial discretization of Eq. 5

[1]In the original federated averaging algorithm, the client C_k provides the locally updated weights $\mathbf{w}_k(t)$ instead of the gradient $\mathbf{g}_k(t)$ (after one or more stochastic gradient descent steps) [23]. We use the form in Eq. 2 to encompass other existing gradient combination models. This original algorithm can be restored by replacing $\mathbf{g}_k(t)$ with $\mathbf{w}_k(t) - \mathbf{w}(t)$ in Eq. 2.

is formulated as a time evolution of H (see [6, 13] for convergence analysis of $\mathbf{L}$ to $-\Delta_M$):

$$\frac{\partial \mathbf{H}}{\partial t} = -\mathbf{L}\mathbf{H} \tag{9}$$

with $\mathbf{H} = [\mathbf{h}_1^\top, \ldots, \mathbf{h}_S^\top]^\top$. This diffuses the rows of $\mathbf{H}$ (equivalently, H) along the manifold M. It should be noted that our diffusion is nonlinear as its generator $\mathbf{L}$ depends on H which is being evolved: Initially, $\mathbf{L}$ is given as a noisy discretization of $-\Delta_M$ (as it is constructed based on H, instead of $\overline{H}$), and $\mathbf{L}$ and $\mathbf{H}$ are jointly denoised during the diffusion process.
Denoising combined gradients via diffusion. We consider a manifold M consisting of normalized gradients of $\mathcal{E}$: $\|\mathbf{g}\|_2 = 1$, $\sum_j [\mathbf{g}]_j = 0$ for $\mathbf{g} \in h(M)$. The corresponding graph Laplacian is constructed using the local gradients $\{\mathbf{g}_k\}_{k=1}^K$ and the combined gradient $\mathbf{q}$: The data matrix $\mathbf{H}$ is obtained by assigning $\mathbf{g}_k$ to $\mathbf{h}_k$ for $1 \le k \le K$ and $\mathbf{q} = \mathbf{h}_S$ with $S = K+1$.

Now applying the explicit Euler time-discretization to the continuous evolution of Eq. 9, we obtain to the following iterative algorithm

$$\begin{aligned} \frac{\mathbf{H}(t+1) - \mathbf{H}(t)}{\delta} &= -\mathbf{L}\mathbf{H}(t) \\ \Leftrightarrow \mathbf{H}(t+1) &= (\mathbf{I} - \delta\mathbf{L})\mathbf{H}(t), \end{aligned} \tag{10}$$

where δ is the discretization step.

This diffusion process simultaneously denoises all gradients, which can be computationally demanding. Since our goal is to robustify the update process of $\mathbf{w}$ (Eq. 3) via the gradient combination $\mathbf{q}$ (Eq. 4), we focus on only diffusing $\mathbf{q}$ given the local gradients $\{\mathbf{g}_k\}$: The evolution of $\mathbf{q}$ is accordingly derived from Eqs. 7 and 10 as

$$\mathbf{q}(t+1) = (1-\delta)\mathbf{q}(t) + \delta \sum_{k=1}^{K} [\mathbf{a}(t)]_k \mathbf{g}_k(t), \tag{11}$$
$$\mathbf{a}(t) = \frac{\tilde{\mathbf{a}}(t)}{\|\tilde{\mathbf{a}}(t)\|_1}, \; [\tilde{\mathbf{a}}(t)]_k = \kappa(\mathbf{g}_k(t-1), \mathbf{q}(t); \sigma^2).$$

Finally, our learning algorithm is obtained by first determining the initial solution $\mathbf{q}(1)$ as a uniform average of the local gradients: $\mathbf{q}(1) = \sum_{k=1}^K [\mathbf{a}(1)]_k \mathbf{g}_i(1)$ with $[\tilde{\mathbf{a}}(1)]_k = 1/K$ and incorporating the resulting diffusion step of $\mathbf{q}$ (Eq. 11) in the update of $\mathbf{w}$ in Eq. 3. By noting that $\mathbf{q}(t)$ is uniquely determined by $\mathbf{a}(t)$ given the local gradients $\{\mathbf{g}_k(t)\}_{k=1}^K$, this casts the $\mathbf{q}$-update step of Eq. 11 into an update rule of the gradient combination weights $\mathbf{a}$:

$$[\mathbf{a}(t+1)]_k = (1-\delta)[\mathbf{a}(t-1)]_k + \delta [\mathbf{a}(t)]_k. \tag{12}$$

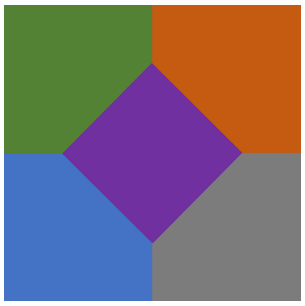
As $\mathbf{a}(t+1)$ is not automatically normalized, we explicitly normalize it before the update of $\mathbf{w}$ (Eq. 3). Algorithm 1 summarizes the training process.

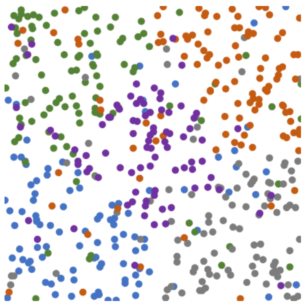
Hyperparameters and complexity. The hyperparameters of our algorithm include the diffusion step δ (Eq. 11) and the kernel width σ^2 (Eq. 8). The parameter σ^2 controls the uniformity of the combination coefficients $\mathbf{a}$: For small σ^2 values, only a small portion of local gradients $\{\mathbf{g}_k\}$ contributes to the combination $\mathbf{q}$ helping suppress noisy gradients, at the expense of potentially neglecting some genuine gradients. Large σ^2 values lead to close to uniform coefficients making more gradients contribute to the combination, possibly including affected gradients. We fix σ^2 as a conservative value of 2 leading to close to uniform $\mathbf{a}$ values for the first 50 iterations of optimization. Thereafter, it is determined as 0.2 which is roughly half the squared mean distance of the pairs of the initial gradients of *CIFAR10* dataset (see Sec. 4). As the explicit Euler scheme in Eq. 11 is not stable for large δ values, we fix δ at a small value of 0.1. This offered fast enough $\mathbf{q}$-update while keeping the evolution stable. For large δ values, when $\mathbf{a}$ is not normalized, $\mathbf{q}$ can quickly diverge to infinity or converge to zero: As $\mathbf{a}$ is normalized in our algorithm, $\mathbf{a}$ actually converges to a vector consisting of zeros except for an entry of value one. In general, tuning these hyperparameters per problem and dataset can improve performance. However, this might require separate validation sets at the server, which can limit the application domain of distributed learning.

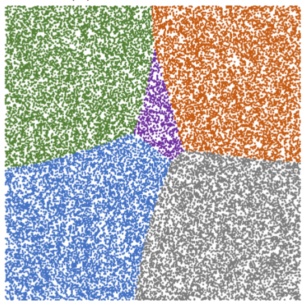
Each step of our algorithm requires updating the gradient combination weights $\mathbf{a}$ (Eq. 12), and calculating the similarity of each local gradient and the combined gradient $\mathbf{q}$ (Eq. 11). Given the local gradients $\{\mathbf{g}_k(t)\}_{k=1}^K$, the time complexity of each step of our algorithm depends linearly on the number K of clients and the size m of a gradient vector (equivalently, the number of weights of the learner f) incur-

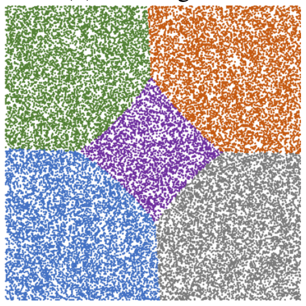
Algorithm 1: Server execution of our distributed learning framework: The client C_k is equipped with a dataset X_k and it provides the corresponding loss gradient $\mathbf{g}_k(t)$ at step t. An unknown subset of the total clients $\{C_k\}_{k=1}^K$ communicate erroneous gradients.

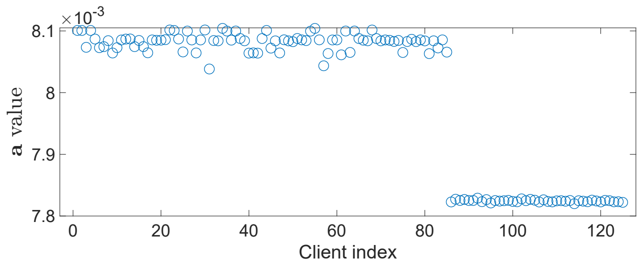
```
Input: Diffusion step size $\delta$ (Eq. 10) and
       kernel width $\sigma^2$ of graph Laplacian (Eq. 8).
Output: Optimized learner parameter $\mathbf{w}^*$.
Initialize the learner parameters $\mathbf{w}(1)$ randomly;
Initialize combination coefficients $\mathbf{a}(1)$ uniformly;
for each round $t = 1, \ldots$ do
    Collects the local gradients $\{\mathbf{g}_k(t)\}_{k=1}^K$;
    Normalize the gradients $\mathbf{g}_k(t) = \frac{\mathbf{g}_k(t)}{\|\mathbf{g}_k(t)\|_2}$;
    Calculate the weighted
      average gradient $\mathbf{q}(t) = \sum_{k=1}^K [\mathbf{a}(t)]_k \mathbf{g}_k(t)$;
    Calculate the kernel evaluations $\tilde{\mathbf{a}}(t)$ (Eq. 11);
    Calculate
      $\mathbf{w}(t+1)$ (Eq. 3) and distribute it to clients;
    Calculate and normalize $\mathbf{a}(t+1)$ (Eq. 12);
end
$\mathbf{w}^* = \mathbf{w}(t)$;
```


(a) Ground truth


(b) Training data


(c) *FedAvg*


(d) Ours



(e) Distribution of the finalized combination weights $\mathbf{a}$.

Figure 1. Our algorithm and *FedAvg* applied to a simple 2D example where the last 40 (out of 125) clients are affected. Our algorithm successfully suppressed the gradients from the affected clients leading to 11.5% accuracy improvements from *FedAvg*.

ring the total complexity of $O(K\times m)$. We terminate the weight diffusion (Eq. 12) when the diversity of local gradients $D_t = \sum_{k=1}^{K}\|\mathbf{g}_k(t-1)-\mathbf{q}(t)\|$ decreases rapidly. For *CIFAR100*, each gradient combination step takes less than a second in a machine with i7-10700K CPU and RTX2080Ti GPU.

An illustrative example. Figure 1 illustrates the learning behavior of our algorithm using a two-dimensional toy example: A training set (Fig. 1(b)) of size 500 is generated as a noisy sample from a ground-truth class map (Fig. 1(a)), which is distributed across 125 clients. At each training iteration, the gradients of 40 clients are replaced by Gaussian random vectors simulating adversarial attacks. Our algorithm successfully suppressed the perturbed gradients as indicated by the small $\mathbf{a}$ values assigned for these affected clients (Fig. 1(e)) and achieved the testing accuracy of 90.81% (Fig. 1(d)). The *FedAvg* was distracted by these random gradients significantly degrading the performance (81.38%; Fig. 1(c)).

Discussion. The gradient combination framework in Eq. 3 is general and it includes other algorithms including Wan and Chen's robust federated learning approach [34] as a special case. In [34], the combination weight $[\mathbf{a}]_k$ is constructed based on the probability of the gradient $\mathbf{g}_k$ being genuine, which is estimated by a separate attention mechanism trained on an annotated dataset (of genuine and affected gradients). This requires a separate dataset at the server and incurs the overhead of transmitting such data to clients to collect the example gradients. Our algorithm can be considered as an alternative that can be applied even when such labeled datasets are not available at the server and when high data communication costs cannot be afforded.

Diffusion processes have been widely used in semi-supervised learning [30, 44], image and surface enhancement [3, 36], clustering and data embedding [5, 33]. Most closely related to our approach is Hein and Maier's manifold denoising algorithm [14] which denoises a given point cloud sampled from a manifold. Unlike [14], our approach employs a probabilistic normalization (Eq. 8) leading to a convex combination of local gradients (Eq. 11) and focuses on diffusing a single combined gradient vector rendering the update rule for the combination weights $\mathbf{a}$ (Eq. 12). Further, our approach uses the explicit Euler time discretization. This leads to a computationally more affordable solution (Eq. 10) than the implicit Euler scheme of [14], as the former does not require solving a large-scale optimization problem per diffusion step. However, this comes at the expense that our update rule is not uniformly stable for any values of the diffusion step $\delta > 0$ (Eqs. 11 and 12) and therefore, we keep it at a small value.

Our algorithm was conceived by simulating diffusion on a data manifold. While this offers an advantage of theoretical rigor inheriting the convergence analysis of diffusion processes and their denoising properties [14, 36], a more intuitive interpretation exists: At step t, the new combined gradient vector $\mathbf{q}(t+1)$ (Eq. 11) can be seen as the minimizer of the following convex energy

$$\mathcal{E}^W(\mathbf{q}) = \frac{1}{2}\sum_{k=1}^{S}[\mathbf{c}(t)]_k\|\mathbf{h}_k(t)-\mathbf{q}\|^2, \tag{13}$$

where for $1 \leq k \leq S-1$, $\mathbf{h}_k(t) = \mathbf{g}_k(t)$, $[\tilde{\mathbf{c}}(t)]_k = \delta[\mathbf{a}(t)]_k$, and $\tilde{\mathbf{c}}(t) = \frac{\mathbf{c}(t)}{\|\mathbf{c}(t)\|}$, and $\mathbf{h}_S(t) = \mathbf{q}(t)$ and $[\tilde{\mathbf{c}}(t)]_S = 1-\delta$. This penalizes the deviation of $\mathbf{q}(t+1)$ from the previous solution $\mathbf{q}(t)$ and the current local gradients $\{\mathbf{g}_k(t)\}$. The contribution $[\mathbf{c}(t)]_k$ of $\mathbf{g}_k(t)$ is controlled based on the *error* $\|\mathbf{q}(t)-\mathbf{g}_k(t-1)\|^2$ incurred at the previous step: The iterative minimization of $\mathcal{E}^W$ tends to assign smaller weights $\{[\mathbf{c}(t)]_k\}$ to the gradients $\{\mathbf{g}_k\}$ that exhibit large errors. This introduces robustness in the gradient aggregation of Eq. 11 as the outliers are gradually ignored during optimization.

Achieving robustness via *reweighting* the contributions of individual error terms has been studied in robust linear regression and compressed sensing [10].

4. Experiments

We evaluated our robust gradient combination algorithm on five classification and three regression datasets.

4.1. Classification

Setting. The *Caltech256* dataset contains 30,607 color images from 257 object and background classes [11]. The *CIFAR10* and *CIFAR100* datasets consist of 60,000 images of 10 and 100 classes, respectively [22]. The *FashionMNIST* dataset consists of 70,000 grayscale images in 10 categories [37]. The *CINIC10* dataset contains 270,000 images selected from ImageNet and CIFAR10 [9].

We simulated *random labeling* perturbations where the ground-truth training labels of the affected clients are replaced by random class labels. In the accompanying supplemental document, we also report the results of experiments with *random gradient* perturbations where the local gradient vectors are replaced by i.i.d. Gaussian random vectors. As our algorithm operates at the server, it is agnostic to the nature of the local datasets and perturbations, and the only information that our algorithm received from each client is the output gradient.

For all datasets, we performed experiments with varying portions of affected clients: Ten to sixty percent of the total clients were affected. For all datasets and portions of affected clients, we ran experiments 10 times with different distributions of training data across clients and averaged the results. For *Caltech256*, *CIFAR10*, *CIFAR100*, *FashionMNIST*, for each experimental run, 20,000 data instances were randomly selected and equally distributed to 100 clients while 10,000 points were used for testing. For *CINIC10*, 180,000 training instances (the union of the training and validation sets proposed by the authors of [9]) were randomly distributed to 1,000 clients, and the remaining 90,000 points were reserved for testing. Here, the sizes of the local datasets across clients varied: The largest local dataset was around 10 times larger than the smallest set.

We used the learner that combines a fixed ResNet101 pretrained on ImageNet and three fully-connected layers.

Baselines. We compared with 1) the naïve application of *FedAvg* which uniformly averages all local gradients. As our algorithm combines these gradients by taking weighted averages instead (Eq. 4), it can be considered as an instance of robust alternatives to the mean. In this context, we compared with 2) the median of local gradients as a popular robust estimator of central statistical tendency (denoted as *Median*) and 3) Pillutla et al.'s geometric median-based approach as another robust average replacement (*GeoMed*) [25]. *GeoMed* requires solving an optimization problem at each gradient combination step (taking around 10 seconds per step). Also, we compared with 4) Turan et al.'s robust gradient-based optimization strategy [32] (*MedTh*). This selects local gradients whose distance from the median gradient is within a threshold. As it is challenging to determine such a threshold value without having access to separate validation sets, we set it in the way that the number of selected gradients becomes the same as the ground-truth number of unaffected clients; Our algorithm did not use this additional information. Lastly, we compared with *FedAvg* applied to only genuine clients (*GTC*). This will serve as the ideal performance upper bounds of all gradient combination algorithms.

Results. Table 1 summarizes the results. Note that the performance of *GTC* degrades with increasing portions of affected clients since the numbers of total genuine training instances accordingly decrease. Federated averaging (*FedAvg*) demonstrated robustness when only moderate numbers (around 10–20%) of clients are affected. However, its performance degraded rapidly as more clients were affected. This tendency was consistently observed across all datasets. While the median is a well-established robust alternative to the mean, combining local gradients based on their median (*Median*) turned out to be only comparable to, or sometimes even significantly worse than the naïve mean (*FedAvg*). *GeoMed* was more effective than *Median*, yet it was noticeably inferior to *FedAvg* in a few cases despite the much higher computational cost. *MedTh* demonstrated competitive performance when affected clients were fewer than 50% of the total clients. However, for larger numbers of affected clients, it also delivered notably worse results than *FedAvg*. Our algorithm demonstrated graceful performance degradation even in these cases, and overall, it yielded (statistically) significantly better results than *FedAvg* on 28 out of 30 cases. Importantly, it was never significantly worse than *FedAvg*.

Throughout the entire experiments on random gradient perturbations (see supplemental document), both *MedTh* and ours significantly outperformed *FedAvg*, constantly achieving the best performance.

Combining *MedTh* and Ours. Our original algorithm estimates the continuous gradient combination weights $\mathbf{a}$. This helps suppress the perturbed gradients but it does not completely rule out their contributions. In the accompanying supplemental document, we demonstrate that when additional validation sets are available, we can achieve this goal and improve performance by combining our algorithm with the thresholding idea of *MedTh*.

4.2. Regression

Setting. We used three benchmark datasets. The *MOCAP* dataset provides 50,000 motion capture data instances [2]. Each instance provides the 3D locations of 62 skeletal

Dataset	Method	Accuracy					
% clients affected		10	20	30	40	50	60
	GTC	76.54 (0.34)	75.96 (0.20)	75.47 (0.19)	74.84 (0.41)	73.83 (0.16)	72.26 (0.50)
	FedAvg	76.29 (0.32)	75.16 (0.35)	74.14 (0.45)	72.29 (0.33)	*70.50 (0.27)*	*66.75 (0.44)*
Caltech256	*Median*	64.80 (0.47)	63.58 (0.97)	61.82 (0.74)	59.15 (0.98)	55.57 (0.85)	48.15 (0.99)
	GeoMed	*76.29 (0.36)*	*75.68 (0.41)*	74.69 (0.44)	71.70 (0.53)	66.49 (0.36)	60.77 (0.82)
	MedTh	76.28 (0.27)	75.59 (0.38)	*74.70 (0.85)*	*72.51 (1.17)*	69.74 (1.04)	46.31 (5.19)
	Ours	**76.39 (0.20)**	**75.95 (0.29)**	**75.06 (0.26)**	**73.72 (0.31)**	**71.90 (0.50)**	**69.25 (0.47)**
	GTC	89.57 (0.11)	89.49 (0.15)	89.44 (0.15)	89.55 (0.20)	89.36 (0.16)	89.22 (0.10)
	FedAvg	89.23 (0.10)	87.34 (0.23)	86.58 (0.15)	85.98 (0.37)	*85.24 (0.41)*	*84.83 (0.37)*
CIFAR10	*Median*	87.21 (0.25)	86.89 (0.26)	86.74 (0.82)	86.30 (1.37)	84.65 (1.19)	82.01 (0.62)
	GeoMed	87.38 (0.47)	87.49 (1.06)	87.14 (1.07)	86.89 (1.83)	84.99 (0.85)	82.46 (0.32)
	MedTh	*89.39 (0.18)*	*89.27 (0.10)*	*89.01 (0.19)*	*88.57 (0.25)*	78.10 (19.90)	31.40 (18.43)
	Ours	**89.51 (0.06)**	**89.43 (0.17)**	**89.22 (0.19)**	**89.14 (0.13)**	**88.95 (0.16)**	**88.85 (0.20)**
	GTC	66.86 (0.23)	66.50 (0.21)	66.10 (0.33)	65.81 (0.30)	65.14 (0.42)	64.41 (0.45)
	FedAvg	66.50 (0.40)	65.95 (0.54)	64.67 (0.28)	63.25 (0.40)	*61.99 (0.32)*	*60.53 (0.33)*
CIFAR100	*Median*	65.76 (0.23)	64.53 (0.29)	62.73 (0.31)	59.51 (0.75)	53.51 (0.85)	47.89 (0.65)
	GeoMed	**67.11 (0.40)**	**66.80 (0.63)**	65.75 (0.69)	*65.20 (0.23)*	58.63 (0.57)	49.79 (1.05)
	MedTh	*66.80 (0.14)*	66.35 (0.29)	*65.78 (0.24)*	64.11 (0.33)	38.34 (1.43)	36.93 (2.52)
	Ours	66.80 (0.23)	*66.41 (0.38)*	**65.87 (0.38)**	**65.44 (0.27)**	**64.80 (0.20)**	**63.56 (0.62)**
	GTC	86.96 (0.18)	86.84 (0.23)	86.84 (0.11)	86.92 (0.19)	86.77 (0.19)	86.77 (0.20)
	FedAvg	86.48 (0.19)	83.12 (0.24)	81.93 (0.35)	81.30 (0.31)	*80.80 (0.50)*	*80.47 (0.26)*
FashionMNIST	*Median*	83.23 (1.44)	82.77 (1.30)	82.68 (1.71)	82.02 (1.57)	80.44 (1.05)	78.40 (0.44)
	GeoMed	83.29 (1.65)	82.37 (0.40)	82.42 (1.72)	81.20 (0.36)	80.41 (0.45)	78.37 (0.30)
	MedTh	*86.75 (0.11)*	*86.55 (0.22)*	*86.27 (0.22)*	*83.57 (4.56)*	76.16 (14.07)	52.65 (23.84)
	Ours	**87.01 (0.23)**	**86.73 (0.16)**	**86.63 (0.27)**	**86.40 (0.51)**	**86.24 (0.16)**	**86.41 (0.27)**
	GTC	79.99 (0.07)	79.96 (0.11)	79.75 (0.12)	79.70 (0.08)	79.66 (0.12)	79.60 (0.15)
	FedAvg	78.20 (0.11)	76.88 (0.21)	76.04 (0.16)	75.41 (0.29)	74.81 (0.27)	*74.38 (0.13)*
CINIC10	*Median*	77.16 (1.59)	76.91 (1.61)	77.03 (1.95)	76.84 (1.33)	73.83 (1.44)	71.04 (0.72)
	GeoMed	78.13 (1.95)	79.01 (1.86)	77.49 (2.61)	*78.92 (1.20)*	*75.64 (1.28)*	71.51 (0.75)
	MedTh	*79.48 (0.11)*	*79.31 (0.15)*	*79.06 (0.10)*	78.57 (0.15)	57.10 (23.87)	42.19 (20.96)
	Ours	**80.03 (0.13)**	**79.95 (0.17)**	**79.85 (0.16)**	**79.64 (0.32)**	**79.19 (0.50)**	**78.69 (0.52)**

Table 1. Results of classification under random labeling perturbations: Mean accuracy and standard deviation (in parenthesis) are shown in % (higher is better). The best and second-best results excluding *GTC* are highlighted with **bold** and *italic*, respectively. For *Median*, *GeoMed*, *MedTh*, and ours, the results of statistical significance test with respect to *FedAvg* (t–test with $\alpha = 0.95$) are shown: Blue and orange respectively represents significantly better and worse results than *FedAvg*.

joints. A body surface mesh model is fitted to these 62 joints and the locations of five end effectors (two hands, two feet, and head) are recorded, constituting 15-dimensional coarse representations. The goal is to restore the 186 (62×3)-dimensional joint locations from the input 15-dimensional coarse representations. We focused on estimating the first 12 attributes (4 out of 62 joints). Our learners consisted of five fully-connected layers. Experiments were repeated 10 times, and for each experimental run, a randomly selected subset of size 40,000 was used for training while the remaining data instances were used for testing. The training sets were distributed across 100 clients where 40% were affected: Each output instance is contaminated with a Gaussian noise whose standard deviation per dimension is twice the standard deviation of the corresponding dimension. The *School* dataset provides exam scores of 15,362 students in 139 UK secondary schools in the years 1985, 1986, and 1987 [4]. Our goal is to predict these exam scores using 27 input features including the year of the exam, gender, and ethnic group. A small subset was selected from each school, constituting a combined test set of size 5,362. The remaining 10,000 instances were distributed to the clients based on the corresponding schools. The *ICVL* hand dataset contains 17,604 depth frames of size 320×240 capturing human hands and the corresponding 3D annotations of 16 joints [31]. Among them, 16,008 frames were used for training while the remaining 1,596 frames were assigned for testing. For each frame, features were extracted using the ImageNet-pretrained ResNet101. Similarly to *MOCAP*, for *School* and *ICVL*, 40% of total local gradients were contaminated by Gaussian noise.

Results. Table 2 summarizes the results. For all datasets, the three robust gradient combination algorithms demonstrated performance gains over *FedAvg*. For *MOCAP*, *GeoMed* was marginally better than *FedAvg* while *MedTh* and ours achieved further significant improvements. For *School*, ours ranked the best while the other two robust algorithms also exhibited marked performance gain. For *ICVL*, *GeoMed* ranked first followed by ours.

5. Conclusions

We have considered the problem of robust distributed learning under adversarial perturbation. Our approach embeds the gradients calculated from local clients into the manifold of normalized vectors and performs diffusion therein. The resulting algorithm is instantiated as a robust average estimator against outliers and it improves the combined gradient without having to require additional labeled datasets at the server or known number of affected clients. On five classification and three regression benchmark datasets, and two scenarios of adversarial perturbations, our algorithm demonstrated significant improvements over baseline federated averaging [23], robust median and geometric median estimators [25], and median-based thresholding approach [32].

Limitations. The application of our approach (as well as the other algorithms considered in our experiments) is limited to cases where individual clients are either entirely affected or not: As our original algorithm is agnostic to data distributions and operations performed within clients, it does not exploit the possibility that only part of each local dataset is perturbed. When access to local datasets is guaranteed, such a scenario could be approached by applying our main gradient update step (Eqs. 11 and 12) in two levels: *Within* each client, it can suppress perturbed data instances by taking each data instance and the corresponding error gradient as a *sub-client*. The resulting locally combined gradients could then be combined at the server as in our main experiments.

Future work and potential negative societal impacts. Our approach contributes to promoting distributed learning, in particular over devices where data acquisition and annotation processes might not be responsibly moderated. This can potentially facilitate the transfer of bias (e.g., age, gender, and ethnicity) present in individual clients to the entire learning process. Therefore, future work should investigate detecting and mitigating the effect of bias in local datasets. Further, future work should assess the performance of our algorithm under different adversarial perturbation scenarios, e.g. when the input images (instead of the labels as in our experiments) are randomly contaminated by noise or systematically altered, and adversarial perturbations are *coordinated* (e.g. by injecting noise correlated across clients). We experimentally demonstrated that our algorithm offers the capability of suppressing affected gradients. Future work should also conduct a theoretical analysis, e.g. on the deviation between the gradients combined by our algorithm and the corresponding ground-truth combinations (generated by *GTC*).

Attr.	*FedAvg*	*GeoMed*	*MedTh*	Ours
		MOCAP		
1	2.05 (0.17)	2.01 (0.06)	*1.75 (0.13)*	**1.74 (0.09)**
2	5.67 (0.05)	5.66 (0.02)	*3.48 (0.10)*	**2.69 (0.08)**
3	18.76 (0.05)	18.62 (0.07)	*9.39 (0.52)*	**4.40 (0.30)**
4	3.43 (0.05)	3.36 (0.05)	*2.75 (0.05)*	**2.55 (0.06)**
5	18.95 (0.08)	18.86 (0.08)	*9.48 (0.53)*	**4.31 (0.35)**
6	1.93 (0.13)	1.84 (0.05)	**1.58 (0.10)**	*1.62 (0.14)*
7	3.96 (0.05)	3.96 (0.05)	*2.89 (0.05)*	**2.46 (0.06)**
8	18.83 (0.09)	18.73 (0.05)	*9.25 (0.49)*	**3.87 (0.30)**
9	2.02 (0.11)	1.92 (0.08)	*1.69 (0.10)*	**1.68 (0.12)**
10	8.78 (0.07)	8.69 (0.06)	*4.41 (0.24)*	**2.47 (0.09)**
11	19.43 (0.05)	19.37 (0.04)	*9.23 (0.56)*	**3.60 (0.21)**
12	2.63 (0.13)	2.55 (0.04)	*2.32 (0.09)*	**2.23 (0.08)**
		School		
1	15.89 (0.36)	*13.36 (0.43)*	13.87 (0.63)	**12.38 (0.41)**
		ICVL		
1	3.61 (0.04)	3.62 (0.11)	3.63 (0.06)	**3.57 (0.05)**
2	3.75 (0.09)	**2.73 (0.09)**	3.17 (0.14)	*3.00 (0.19)*
3	16.40 (0.08)	**4.53 (0.08)**	9.09 (0.67)	*7.02 (1.60)*
4	*3.46 (0.02)*	3.56 (0.08)	**3.43 (0.09)**	3.49 (0.04)
5	4.03 (0.04)	**2.95 (0.12)**	3.58 (0.11)	*3.36 (0.14)*
6	16.19 (0.04)	**4.50 (0.10)**	8.98 (0.60)	*6.90 (1.60)*
7	*3.77 (0.02)*	3.94 (0.11)	**3.70 (0.04)**	3.79 (0.08)
8	3.91 (0.10)	**3.32 (0.25)**	3.56 (0.11)	*3.46 (0.09)*
9	15.33 (0.08)	**4.56 (0.10)**	8.54 (0.63)	*6.65 (1.43)*
10	4.52 (0.05)	4.50 (0.05)	**4.39 (0.03)**	*4.42 (0.10)*
11	*3.69 (0.05)*	**3.64 (0.29)**	3.73 (0.12)	3.69 (0.06)
12	15.18 (0.09)	**5.02 (0.04)**	8.74 (0.56)	*7.01 (1.33)*

Table 2. Regression results: Mean error rate and standard deviation in parenthesis (lower is better). The best and second-best results are highlighted with **bold** and *italic*, respectively. Statistically significantly better and worse results than *FedAvg* are highlighted in blue and orange, respectively.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant (No. 2021R1A2C2012195), Institute of Information & Communications Technology Planning & Evaluation (IITP) grant (2021–0–00537, Visual Common Sense Through Self-supervised Learning for Restoration of Invisible Parts in Images), and IITP grant (2020–0–01336, Artificial Intelligence Graduate School Program, UNIST), all funded by the Korea government (MSIT).

References

[1] Dan Alistarh, Zeyuan Allen-Zhu, and Jerry Li. Byzantine stochastic gradient descent. In *NeurIPS*, 2018. 2

[2] A. Baak, M. Müller, G. Bharaj, H.-P. Seidel, and Christian Theobalt. A data-driven approach for real-time full body pose reconstruction from a depth camera. In *ICCV*, pages 1092–1099, 2011. 6

[3] Chandrajit L. Bajaj and Guoliang Xu. Anisotropic diffusion of surfaces and functions on surfaces. *ACM Trans. Graphics*, 22(1):4–32, 2003. 5

[4] B. Bakker and T. Heskes. Task clustering and gating for Bayesian multitask learning. *JMLR*, 4, 2003. 7

[5] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15:1373–1396, 2003. 5

[6] Mikhail Belkin and Partha Niyogi. Towards a theoretical foundation for Laplacian-based manifold methods. *Journal of Computer and System Sciences*, 74(8):1289–1308, 2005. 4

[7] Aurélien Bellet, Rachid Guerraoui, Mahsa Taziki, and Marc Tommasi. Personalized and private peer-to-peer machine learning. In *AISTATS*, 2018. 2

[8] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. In *ICML*, 2019. 2

[9] Luke N. Darlow, Elliot J. Crowley, Antreas Antoniou, and Amos J. Storkey. CINIC-10 is not ImageNet or CIFAR-10. In *arXiv:1810.03505*, 2018. 6

[10] Ingrid Daubechies, Ronald DeVore, Massimo Fornasier, and C. Sinan Güntürk. Iteratively reweighted least squares minimization for sparse recovery. *Communications on Pure and Applied Mathematics*, 63(1):1–38, 2009. 6

[11] Greg Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. Technical report, California Institute of Technology, 2007. 6

[12] Yufei Han and Xiangliang Zhang. Robust federated learning via collaborative machine teaching. In *AAAI*, 2019. 1, 2

[13] Matthias Hein, Jean-Yves Audibert, and Ulrike von Luxburg. From graphs to manifolds - weak and strong pointwise consistency of graph Laplacians. In *COLT*, pages 470–485, 2005. 4

[14] Matthias Hein and Markus Maier. Manifold denoising. In *NIPS*, pages 561–568, 2007. 3, 5

[15] Zeou Hu, Kiarash Shaloudegi, Guojun Zhang, and Yaoliang Yu. FedMGDA+: federated learning meets multi-objective optimization. In *arXiv:2006.11489*, 2020. 2

[16] J. Jost. *Riemannian Geometry and Geometric Analysis*. Springer, New York, 2011. 3

[17] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konecný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and open problems in federated learning. *Foundations and Trends in Machine Learning*, 14(1–2):1–210, 2021. 2

[18] Mikhail Khodak, Maria-Florina Balcan, and Ameet Talwalkar. Adaptive gradient-based meta-learning methods. In *NeurIPS*, 2019. 2

[19] Anastasia Koloskova, Tao Lin, Sebastian U. Stich, and Martin Jaggi. Decentralized deep learning with arbitrary communication compression. In *ICLR*, 2020. 2

[20] Anastasia Koloskova, Nicolas Loizou, Sadra Boreiri, Martin Jaggi, and Sebastian Stich. A unified theory of decentralized SGD with changing topology and local updates. In *ICML*, 2020. 2

[21] Anastasia Koloskova, Sebastian Stich, and Martin Jaggi. Decentralized stochastic optimization and gossip algorithms with compressed communication. In *ICML*, 2019. 2

[22] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. 6

[23] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, 2017. 1, 2, 3, 8

[24] Mehryar Mohri, Gary Sivek, and Ananda Theertha Suresh. Agnostic federated learning. In *ICML*, 2019. 2

[25] Krishna Pillutla, Sham M. Kakade, and Zaid Harchaoui. Robust aggregation for federated learning. In *arXiv:1912.13445*, 2019. 2, 6, 8

[26] S. Rosenberg. *The Laplacian on a Riemannian Manifold*. Cambridge University Press, 1997. 3

[27] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet Talwalkar. Federated multi-task learning. In *NeurIPS*, 2017. 1, 2

[28] Lili Su and Jiaming Xu. Securing distributed gradient descent in high dimensional statistical learning. In *ACM Measurement Analysis and Computer System*, 2019. 2

[29] Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H. Brendan McMahan. Can you really backdoor federated learning? In *arXiv:1911.07963*, 2019. 2

[30] Arthur D. Szlam, Mauro Maggioni, and Ronald R. Coifman. Regularization on graphs with function-adapted diffusion processes. *JMLR*, 9:1711–1739, 2008. 5

[31] Danhang Tang, Hyung Jin Chang, Alykhan Tejani, and Tae-Kyun Kim. Latent regression forest: structured estimation of 3D articulated hand posture. In *CVPR*, 2014. 7

[32] Berkay Turan, César A. Uribe, Hoi-To Wai, and Mahnoosh Alizadeh. Robust distributed optimization with randomly corrupted gradients. In *arXiv:2106.14956*, 2021. 1, 2, 6, 8

[33] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007. 5

[34] Ching Pui Wan and Qifeng Chen. Robust federated learning with attack-adaptive aggregation. In *IJCAI Workshop on Federated and Transfer Learning for Data Sparsity and Confidentiality*, 2021. 1, 2, 5

[35] Jianyu Wang, Anit Kumar Sahu, Zhouyi Yang, Gauri Joshi, and Soummya Kar. MATCHA: speeding up decentralized SGD via matching decomposition sampling. In *arXiv:1905.09435*, 2019. 2

[36] J. Weickert. *Anisotropic Diffusion in Image Processing*. ECMI Series, Teubner-Verlag, Stuttgart, 1998. 3, 5

[37] Han Xiao, Kashif Rasul, and Roland Vollgraf. FashionMNIST: a novel image dataset for benchmarking machine learning algorithms. In *arXiv:1708.07747*, 2017. 6

[38] Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. Phocas: dimensional Byzantine-resilient stochastic gradient descent. In *arXiv:1805.09682*, 2018. 2

[39] Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. Fall of empires: breaking Byzantine-tolerant SGD by inner product manipulation. In *UAI*, 2019. 2

[40] Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. Zeno: distributed stochastic gradient descent with suspicion-based fault-tolerance. In *ICML*, 2019. 2

[41] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: concept and applications. *ACM Trans. Intelligent Systems and Technology*, 10(2):12:1–19, 2019. 1

[42] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. Byzantine-robust distributed learning: towards optimal statistical rates. In *ICML*, pages 5650–5659, 2018. 2

[43] Chen Yu, Hanlin Tang, Cedric Renggli, Simon Kassing, Ankit Singla, Dan Alistarh, Ce Zhang, and Ji Liu. Distributed learning over unreliable networks. In *ICML*, 2019. 2

[44] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *ICML*, pages 912–919, 2003. 5