When Device Delays Meet Data Heterogeneity in Federated AIoT Applications

Haoming Wang
University of Pittsburgh
USA
hw.wang@pitt.edu

Wei Gao
University of Pittsburgh
USA
weigao@pitt.edu

Abstract

Federated AIoT uses distributed data on IoT devices to train AI models. However, in practical AIoT systems, heterogeneous devices cause data heterogeneity and varying amounts of device staleness, which can reduce model performance or increase federated training time. When addressing the impact of device delays, existing FL frameworks improperly consider it as independent from data heterogeneity. In this paper, we explore a scenario where device delays and data heterogeneity are closely correlated, and propose FedDC, a new technique to mitigate the impact of device delays in such cases. Our basic idea is to use gradient inversion to learn knowledge about device's local data distribution and use such knowledge to compensate the impact of device delays on devices' model updates. Experiment results on heterogeneous IoT devices show that FedDC can improve the FL performance by 34% with high amounts of device delays, without impairing the devices' local data privacy.

CCS Concepts

• **Computer systems organization → Embedded and cyber-physical systems**; • **Computing methodologies → Artificial intelligence**.

Keywords

Artificial Intelligence of Things, Federated Learning, Device Delays, Data Heterogeneity

ACM Reference Format:
Haoming Wang and Wei Gao. 2025. When Device Delays Meet Data Heterogeneity in Federated AIoT Applications. In *The 31st Annual International Conference on Mobile Computing and Networking (ACM MOBICOM '25), November 4–8, 2025, Hong Kong, China.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3680207.3723481

ACM MOBICOM '25, Hong Kong, China

ACM ISBN 979-8-4007-1129-9/2025/11
https://doi.org/10.1145/3680207.3723481

1 Introduction

The Artificial Intelligence of Things (AIoT) [2, 20, 61] perceive the heterogeneous data produced by IoT devices using AI models, enabling many intelligent applications including smart health [7, 41, 60], autonomous driving [3, 11, 21] and smart cities [36, 37]. Since data in AIoT applications is generated at distributed sources but cannot be processed at a central server due to large sizes [22] and privacy concerns [69], Federated Learning (FL) [35] is usually used to train AI models [18, 24]. In FL, each IoT device receives the global model from the server and trains the model using local data, and trained model updates are transmitted to and aggregated at the server to update the global model. This procedure iteratively repeats until the global model converges.

FL in AIoT faces two major challenges, namely *data heterogeneity* and *device delays*. Data heterogeneity exists as data on different IoT devices is not independent and identically Distributed (i.i.d.) due to diverse user behaviors or environmental contexts [19, 27], and leads to biased model updates from devices that degrades the global model's performance [23, 30, 64]. Device delays, on the other hand, refer to the extra time for the server to receive the model update from a slow IoT device, and can be caused by various reasons in AIoT, such as device's insufficient compute power, excessive amounts of local compute workloads, and disruptions in communication between devices and the server. Such delays could cause significant slowdown of FL training [25, 44].

Current FL frameworks use different ways to tackle device delays. For example, conventional synchronous FL discards delayed model updates if the delay is too long [29], and asynchronous FL [9, 56] applies lower weights to delayed model updates at server's aggregation [10, 64]. However, most approaches consider device delays as *independent* from data heterogeneity, and they cannot be applied to many AIoT applications where device delays and data heterogeneity are closely correlated, i.e., data of a certain class or with specific features may only be available on some slow devices. For example, as shown in Figure 1, in human activity recognition (HAR) [45, 52], users doing certain outdoor activities (e.g., hiking, field repairs, etc) may always have limited network connectivity during these activities, and model updates regarding these activities will always be delayed. Similar cases

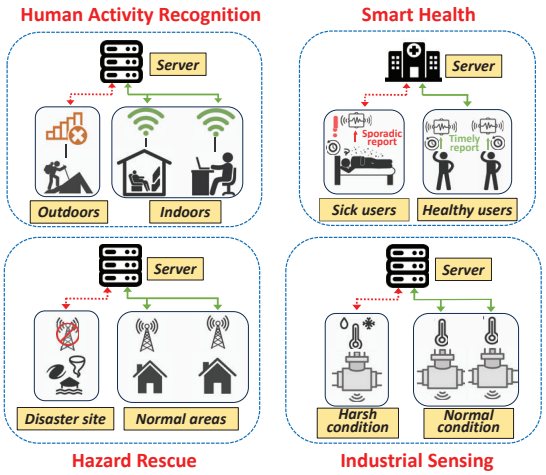


Figure 1: Federated AIoT applications where device delays and data heterogeneity are correlated

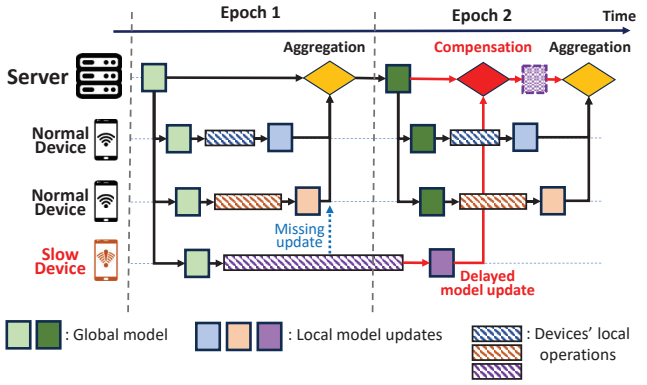


Figure 2: Delayed model updates will be compensated and then aggregated to the global model. The devices' local operations include local training and transmission of the trained model update to the server.

are also found in hazard rescue operations [1] and industrial manufacturing sites, where IoT devices operate in harsh conditions [4]. In these applications, discarding the delayed model updates or reducing their contributions at server's aggregation will exclude knowledge about unique data from the global model, resulting in significant model bias.

Instead, knowledge in delayed model updates should be fully aggregated into the global model. To do this without affecting FL performance, as shown in Figure 2, a better solution is to estimate and compensate the impact of device delays on these updates before aggregation. Since the model update at an IoT device is computed by training the global model with device's local data, when device delay is small (e.g., within one FL epoch), we can estimate the impact of device delays by applying Taylor expansion on the gradient of delayed model update and use its first-order term as the estimator [65]. However, device delays in AIoT could usually be large or even unbounded [67], especially in many adverse application contexts as shown in Figure 1. In these cases, the error of such estimation will be significantly enlarged.

To accurately estimate and compensate the impact of unbounded device delay in AIoT, knowledge about the slow IoT device's local data must be sufficiently exploited. Based on this vision, in this paper we present *Delay Compensator in FL (FedDC)*, a new FL technique that uses gradient inversion [68] to learn knowledge about the slow device's local data distribution and further use such knowledge at the server to mimic the slow device's local training with the global model. In this way, since the device's local training procedure in FL is independent from the device delay, our approach can ensure accurate estimation on the impact of device delay in any amount. In practice, the error of such compensation will gradually increase as FL training progresses, and we adaptively decide when to end such compensation and switch back to vanilla FL according to the specific training progress.

To our best knowledge, our work is the first that focuses on the correlation between device delays and data heterogeneity in FL, which is critical to federated AIoT applications. In particular, our approach has the following advantages:

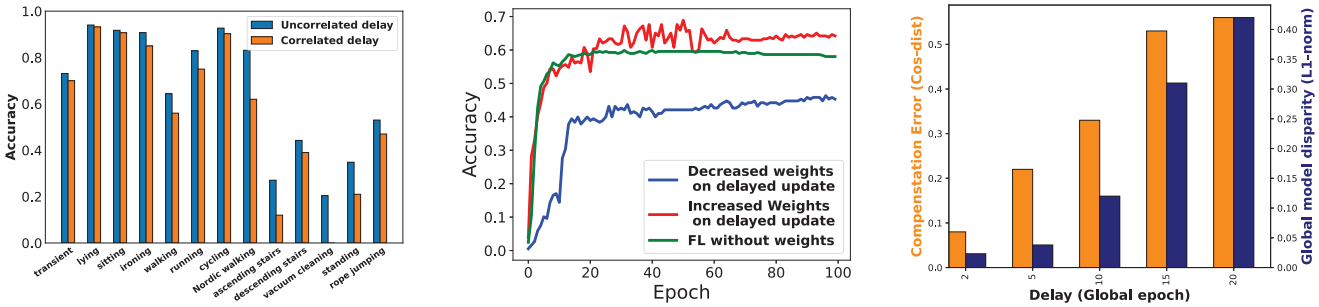
- FedDC retains the local training procedure at IoT devices in FL to be completely unchanged, but executes all extra computations at the FL server. It hence does not incur any additional computation or communication overhead at IoT devices.
- FedDC does not require any auxiliary dataset nor the IoT devices' local models to be fully trained, and can hence be applied to any stage of the FL procedure.
- In FedDC, the server is unable to recover any original samples or labels of IoT devices' local data, and hence completely avoids impairing the devices' data privacy.

We implemented FedDC on multiple types of IoT devices including different models of smartphones, Raspberry Pi 4B and NVidia Jetson Nano, and evaluated its performance on 3 real-world AIoT datasets. From our experiment results, we have the following conclusions:

- FedDC is *accurate*. Compared with the baselines [8, 10, 15, 35, 65], it can improve the trained model's accuracy by up to 34%, even with high amounts of correlated device delays and data heterogeneity.
- FedDC is *adaptive*. It exhibits significant improvements of FL performance with different AIoT datasets, neural network models and experiment settings.
- FedDC is *lightweight*. It incurs the minimum amount of extra computations at the server, which does not result in any delay on FL training.

2 Background and Motivation

In this section, we first demonstrate the FL performance drop in AIoT when device delays meet data heterogeneity,



(a) The impact on the global model's accuracy

(b) The effectiveness of applying weights on delayed model updates

(c) The error of compensating the impact of device delays with first-order estimation

Figure 3: FL performance when device delays are correlated with data heterogeneity

and experimentally verify the ineffectiveness of existing approaches to tackling such FL performance drop. Such ineffectiveness, then, motivates us to instead learn knowledge about slow IoT device's local data by leveraging gradient inversion, to ensure that such unique knowledge can be fully incorporated into the global model.

2.1 FL Performance when Device Delays meet Data Heterogeneity

Both device delays and data heterogeneity affect FL performance in AIoT. However, when they are correlated, device delays will only affect specific data classes that are uniquely available on slow devices, as shown in Figure 1, and the global model's accuracy degradation in these classes will be more significant. To demonstrate this, we conducted preliminary experiments on 13 IoT devices that use FL to train a 3-layer MLP classification model on the PAMAP2 dataset [43], which contains 13 classes of human activities. FedAvg [35] is used for server's aggregation. This dataset naturally incorporates data heterogeneity, such that each device should only contain data in one class [43]. When data heterogeneity is uncorrelated with device delays, in each FL training epoch we randomly select one device to apply a delay of 10 epochs. Otherwise when data heterogeneity is correlated to device delays, we randomly select one data class and assign delays only to devices with data samples in that class. Results in Figure 3(a) show that, in this case, the accuracy on selected data class significantly drops, motivating the need of well designed techniques to address such model accuracy drop.

The most commonly used technique is to reduce the weights of delayed model updates in aggregation [10, 29, 64] . However, when device delays are correlated to data heterogeneity, although using lower weights on delayed model updates reduces the errors applied on the global model, it also prevents the unique knowledge contained in these updates from being sufficiently aggregated to the global model, leading to further model accuracy drop as shown in Figure 3(b). Reversely, if higher weights are applied to delayed model updates, it improves the model accuracy in the data classes affected by device delays, but also amplifies the error in the global model and hence reduces the overall model accuracy in other data classes. For example, in our experiments, the overall classification accuracy over the 13 classes of human activities dropped from 68.6% to 65.2%.

These results show that modifying weights on delayed model updates cannot address correlated device delays and data heterogeneity. Instead, a better solution is to estimate and compensate the impact caused by device delays on model updates. The existing method of estimation is to apply Taylor expansion on the gradient of delayed model update, and to use its first-order term as the estimator by assuming the device delay is always sufficiently small [15, 65]. For a model update $g(w_{t-\tau})$ at time t where τ is the amount of device delay and $w_{t-\tau}$ indicates the global model used to compute the update, the compensated update is calculated as

$$\hat{g}(w_t) \approx g(w_{t-\tau}) + \nabla g(w_{t-\tau})(w_t - w_{t-\tau}), \tag{1}$$

and the Hessian matrix $\nabla g(w_{t-\tau})$ is approximated as

$$\nabla g(w_{t-\tau}) \approx \lambda \cdot g(w_{t-\tau}) \odot g(w_{t-\tau}) \tag{2}$$

where $\odot$ means element-wise multiplication and λ is a hyperparameter. However, the error of such first-order compensation, i.e., the high-order terms in Taylor expansion, increases with the amount of device delay (τ). To verify this, we use the same experiment settings as above, and results in Figure 3(c) show that when we use the compensated model update in aggregation, the discrepancy caused in the global model significantly enlarges when device delay grows. These results motivate designing a better method that compensates the impact caused by device delays of any amount.

2.2 Gradient inversion

Our proposed approach leverages gradient inversion [68], which was designed to learn knowledge about the training data from the gradient of a trained model, by minimizing

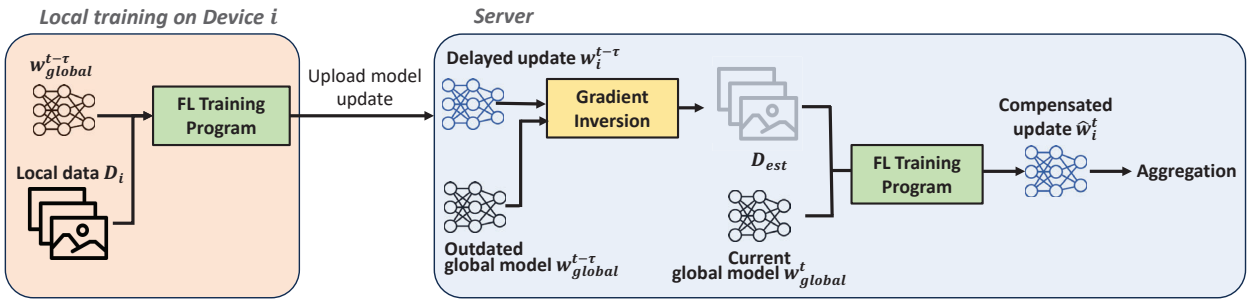


Figure 4: Overview of FedDC design

the difference between the trained model's gradient and the gradient computed from the estimated data. More specifically, it solves the following optimization problem:

$$(x'^*, y'^*) = \arg\min_{(x',y')} \| \frac{\partial L[(x', y'); w^{t-1}]}{\partial w^{t-1}} - g^t \|_2^2, \quad (3)$$

where x' and y' indicate the estimated training data samples and data labels, respectively, w^{t-1} is the trained model, $L[\cdot]$ is model's loss function, and g^t is the gradient calculated with the training data and w^{t-1}. This problem can be solved using gradient descent to iteratively update (x', y').

The effectiveness of such learning is directly related to the volume of training data, and is usually limited to estimating a small number of training data samples [13, 63, 68]. The quality of estimated data is sensitive to extra disturbances applied to the model's gradient, such as sparsification and random noise [68]. These limitations of gradient inversion motivate us to seek better ways of learning the knowledge about slow IoT devices' local data, but also highlight the possibility of preventing such learning from impairing the devices' local data privacy.

3 Overview

As shown in Figure 4, the primary rationale of FedDC design is to keep the IoT device's local FL procedure completely unchanged to avoid any extra communication and computation overhead, but to compensate the impact of device delays at the server. Upon receiving a delayed model update from a device, since this update was calculated at the device using an outdated global model, the server correspondingly caches the outdated global model from previous epochs, and uses this outdated global model to learn knowledge about the device's local data via gradient inversion. Such knowledge is then used to compensate the error in the model update due to device delays, and the compensated model update is aggregated into the current global model.

3.1 Compensating Device Delays

We consider a *semi-asynchronous* FL scenario where some normal IoT devices are not affected by device delays and follow synchronous FL, and some other slow devices send model updates asynchronously [8]: at time t, the server receives a model update that is delayed by τ:

$$w_i^{t-\tau} = LocalUpdate(w_{global}^{t-\tau}; D_i), \quad (4)$$

which is computed by device i using its local data D_i to train an outdated global model $w_{global}^{t-\tau}$. If the device delay does not exist, the corresponding model update should be

$$w_i^t = LocalUpdate(w_{global}^t; D_i). \quad (5)$$

The primary objective of FedDC, hence, is to estimate and compensate the impact of device delays caused in $w_i^{t-\tau}$, by approximating w_i^t. Our primary approach is that the server applies gradient inversion described in Eq. (3) on $w_i^{t-\tau}$, to learn an intermediate dataset D_{est} that approximates the device i's local data D_i, and then use D_{est} to retrain the server's current global model (w_{global}^t). In this way, we expect that the outcome of such retraining can approximate w_i^t.

Specifically, we randomly initialize each data sample and label in D_{est}, and then iteratively update D_{est} by minimizing

$$Dist[LocalUpdate(w_{global}^{t-\tau}; D_{est}), w_i^{t-\tau}], \quad (6)$$

using gradient descent, where $Dist[\cdot]$ is a metric to evaluate how much $w_i^{t-\tau}$ changes if retrained using D_{est}. In FL, a client's model update comprises multiple local training steps instead of a single gradient. Hence, to use gradient inversion, we substitute the single gradient computed from D_{est} in Eq. (3) with the outcome of the device's local training using D_{est}.

To decide the appropriate choice of the metric $Dist[\cdot]$, we evaluate the global model's accuracy when using different metrics in Eq. (6), with the same experiment setting as described in Section 2.1. As shown in Figure 5, using L1-norm metric provides the most reliable performance, while using either L2-norm or cosine distance metric result in significance FL performance drop or instability in training. The basic reason is that using L1-norm metric ensures unbiased sampling in gradient inversion, and we will use L1-norm as the choice of metric in the rest of this paper.

However, according to [31, 62], it is difficult for gradient inversion to precisely reconstruct individual samples of the training data. Instead, in FedDC we ensure that the estimated D_{est} provides accurate knowledge about the device's local

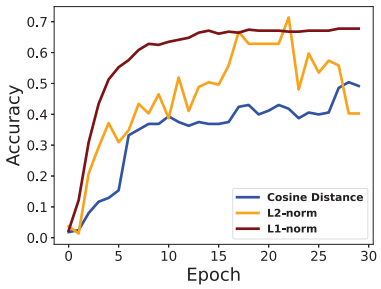
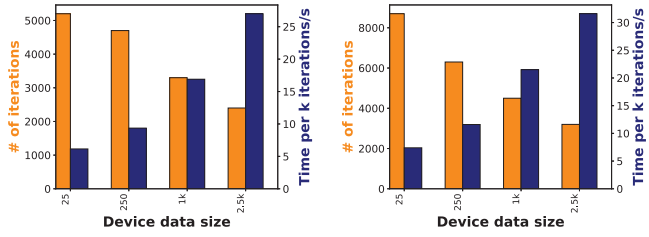


Figure 5: The global model's accuracy when using different metrics for $Dist[\cdot]$ in Eq. (6)

data distribution. In Section 4.1, we will further show that, with such knowledge about data distribution, FedDC can still ensure close approximation to w_i^t.

FedDC's capability of such approximation is also related to the size of D_{est}, which decides both the converging loss and computing cost of gradient inversion. In Section 4.2, we provide more details about how to balance between these two aspects by deciding the proper size of D_{est}.

Furthermore, the effectiveness of FedDC's approximation on improving the global model's accuracy will diminish, as the FL training progresses and the global model's change over different training epochs becomes smaller. As a result, the approximation approach in FedDC would result in larger error in the late stage of FL training, and we need to adaptively switch back to vanilla FL as needed. More details of such switch are in Section 4.3.



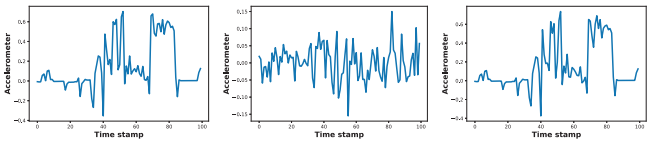
(a) PAMAP2 [43] (b) ExtraSensory [52]

Figure 6: Iterations and time needed for gradient inversion with different AIoT datasets

3.2 Reducing the Computation Overhead

Gradient inversion is known to be computationally expensive, due to its iterative optimizations as specified in Eq. (3). As shown in Figure 6, for different AIoT datasets, we will need at lead 2,000 iterations for the loss of gradient inversion to converge, and the time needed for each iteration ranges between 7.4ms and 31.6ms depending on the size of D_{est}, when running on a NVidia A5000 GPU with 24GB memory. With Such high computing overhead, hence, will become a major burden for the FL server.

To reduce such computation overhead, our approach is to reduce the complexity of optimization objective in Eq. (3) via sparsification. Existing studies showed that model updates in FL are highly sparse [17, 34], and FL performance is mainly determined by <5% of the largest gradients in model updates. Hence, gradient inversion only needs to perform on these gradients with high magnitudes, leading to significant overhead reduction without affecting the accuracy of approximation. Further, we also reduce the scope to which gradient inversion is applied, based on the difference of delay compensation in different FL training epochs. More details of such reduction of computation overhead are in Section 5.



(a) Original IMU data (b) Estimated 100 samples (c) Estimated 1 sample

Figure 7: Examples of comparing the IMU data estimated by using gradient inversion with the original IMU data in the ExtraSensory dataset [52]

3.3 Protecting IoT Devices' Data Privacy

One major concern of using gradient inversion for delay compensation in FL is the possible leakage of IoT devices' local data privacy, as FedDC uses gradient inversion to learn knowledge about devices' local data. To verify this risk, we checked the time-series data samples in D_{est} that gradient inversion estimated from the ExtraSensory dataset [52][1]. Results in Figure 7(b), as an example, show that in a typical FL setting where each IoT device has a large volume of local data samples, it is hard to correlate the estimated data samples in D_{est} with these many samples in the original data. However, in another setting shown in Figure 7(c) where data on each IoT device only contains few data samples, these samples can be precisely recovered by gradient inversion.

To eliminate such risk of privacy leakage, we adopt the similar sparsification techniques to mitigate the capability of gradient inversion on recovering the IoT device's original data samples. Our results in Section 6 show that, even in the most challenging setting, we can prevent the data samples estimated by gradient inversion from being recognizable by both human eyes and a neural network classifier.

4 Approximating Model Updates without Device Delays

In this section, we provide more details about FedDC's approach to learning the intermediate dataset D_{est} and using D_{est} to approximate model updates without device delays.

[1]We noted that in most AIoT applications, the data produced at IoT devices exhibits as time series. Typical examples of such data include humans' body motion, location, heart rate and surrounding environmental conditions, as contained in the PAMAP2 [43] and ExtraSensory [52] datasets.

4.1 Towards Accurate Approximation

We first validate that the loss surface in model's weight space, when computed using D_{est}, can closely resemble that computed using D_i, thereby enabling the computation of a similar gradient. More specifically, we visualize the loss surface by projecting it onto a 2D space, as described in [28]. The results in Figure 8, with a 3-layer MLP model being trained on the PAMAP2 dataset [43], indicate that the loss surface computed using D_{est} closely mirrors that obtained using D_i on the current global model w^t_{global}, and such difference is as small as 0.05 when measured as the cosine distance.

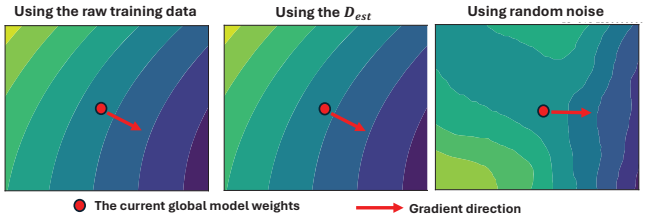


Figure 8: Visualization of the loss surface

Based on such similarity, when we use D_{est} to approximate w^i_t as described in Eq. (6), we compare its approximation error with the first-order compensation [15, 65] described in Eq. (1), by measuring the difference between the approximated w^i_t and the corresponding ground truth without device delays. With that same experiment settings as described above, results in Figure 9 show that FedDC slightly outperforms first-order compensation when the device delay is small. However, as the delay increases, FedDC achieves significantly lower errors. Notably, when the device delay is 20 epochs, FedDC reduces the estimation error by 40%.

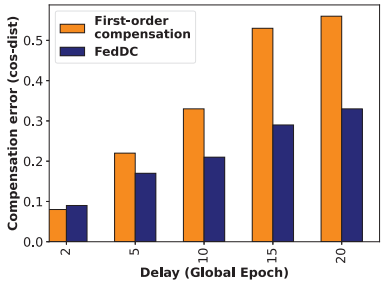


Figure 9: Comparing the approximation error of FedDC with that of first-order compensation [15, 65], measured in cosine distance

4.2 Deciding the size of D_{est}

Gradient inversion is equivalent to data resampling from the IoT device's local data distribution. Hence, a sufficiently large D_{est} is necessary to ensure unbiased data sampling and minimize the loss of gradient inversion. However, if D_{est} is too large, the computational time per iteration becomes unnecessarily high.

The intuitive solution is to let D_{est} be as large as the IoT device's local training data D_i [58]. However, in AIoT applications, devices' local data is usually produced continuously at high frequencies [26, 47], resulting in large volumes of data but high similarity between consecutive data samples. Such similarity enables opportunities to reduce the size of D_{est} without affecting the convergence of gradient inversion. We further investigated such opportunities with a 3-layer MLP model and the PAMAP2 dataset [43]. As shown in Table 1, when the size of D_{est} is 1/16 of that of D_i, we can effectively minimize the loss of gradient inversion at convergence, without incurring extra computation overhead. Further increasing the size of D_{est} results in very small reduction of the loss but significantly increases the computing overhead.

Size	1/128	1/64	1/32	1/16	1/4	1/2	1
Time (s)	5.3	9	27	62	119	208	391
GI loss	8.7	4.4	0.55	0.39	0.37	0.37	0.36

Table 1: The converging loss and computing cost of gradient inversion (GI) with different sizes of D_{est}, measured as the ratio of the IoT device's local data size

4.3 Adaptively Switch back to Vanilla FL

As shown in Table 1, even with a large D_{est}, the loss of gradient inversion at convergence can never be reduced to zero, meaning that the approximation in FedDC can never be 100% accurate but will always contain errors. On the other hand, as the FL training progresses and the global model converges, the difference between the current and outdated global models reduces, eventually to zero. This implies that in the late stage of FL training, FedDC's approximation and compensation to device delays would result in more errors to the global model, as shown by our experiment results in Figure 10 using the same settings as above.

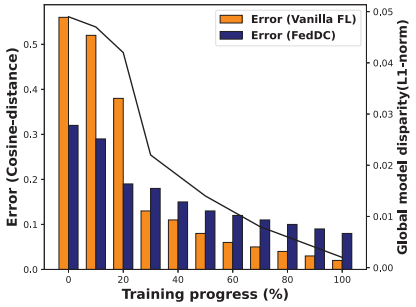


Figure 10: FedDC's error of approximation at different stages of FL training

In this case, FedDC should adaptively switch back to vanilla FL in the late stage of FL training. The major challenge of deciding when to switch is that at any time t in FL training, as long as the device delay is still present, the ground truth

of the model update unaffected by device delay (i.e., w_i^t) is unknown, and it is hence difficult for the server to estimate the current approximation error of FedDC. To address this challenge, we observe that, according to Eq. (5), w_t^i is calculated by device i using the current global model w_{global}^t, and hence it will arrive at the server at a later time, namely $t+\tau'$. Therefore, we can empirically decide to switch at time $t+\tau'$, if by then we find that using FedDC's approximation to w_i^t results in more errors compared to using the actually received w_i^t. Details of such switching are in Algorithm 1.

Algorithm 1 Decision on whether to switch

```
initialize Switch ← False
for each global epoch t + τ' do
    if having ground truth (ŵ_i^t) of the previous estimation (w_i^t) then
        if Switch = False then
            retrieve previous estimations w_i^t, w_i^{t−τ}
            E_FedDC(t) ← Dist(ŵ_i^t, w_i^t)
            E_vannila(t) ← Dist(w_i^{t−τ}, w_i^t)
            if E_FedDC(t) > E_vannila(t) then
                Switch ← True
    if Switch = True then
        Compensate for delayed updates in this epoch
    else
        Directly aggregate the delayed updates
```

5 Compute Efficiency at FL Server

5.1 Sparsification in Gradient Inversion

To reduce the computation overhead caused by gradient inversion on the FL server, our primary approach is gradient sparsification that only involves the important gradients with large magnitudes into gradient inversion. As shown in Figure 11, by only involving the top 5% of gradients, we can reduce 55% of iterations in gradient inversion, with only slight increase in the error of approximating model updates without device delay. Further increasing the sparisification rate to 99%, on the other hand, will lead to significant increase of the approximation error.

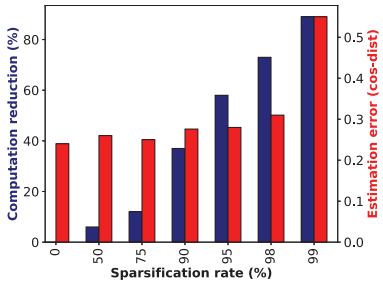


Figure 11: Computation reduction and approximation error under different rates of sparsification

5.2 Selective Computation

Another possibility to further reduce the computation overhead is to apply gradient inversion in a limited scope of compensating model updates. First, if the local data on IoT devices only exhibit small changes over different FL epochs, we do not need to estimate D_{est} every time from scratch, but can instead initialize D_{est} with the results in previous epochs, hence reducing the number of iterations required for gradient inversion. As shown in Figure 12, if the devices' local data is completely unchanged, we can further reduce the number of iterations by 91%. Even if the devices' local data changes by 50%, the extra reduction can still be 82%.

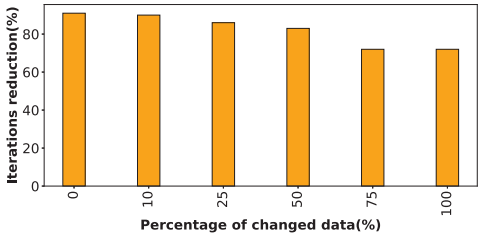


Figure 12: The number of iterations reduced by initializing D_{est} with the results in previous epochs

Similarly, when the IoT device's local data exhibits small changes over time, the delayed model updates received from the same IoT device in different training epochs could also overlap and share much knowledge in common. In this case, we only need to compensate the delayed model updates that contain unique knowledge. We define such uniqueness using the difference in updates' gradient direction with others and quantify it via cosine distance, such that

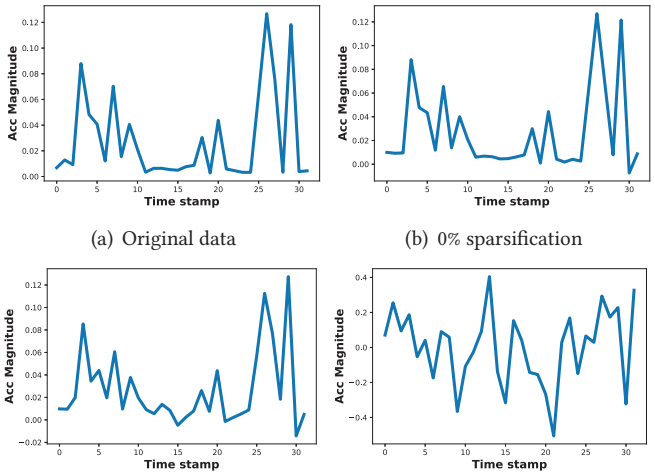
$$D_c(w_i^{t_1}, w_i^{t_2}) = 1 - w_i^{t_1} \cdot w_i^{t_2} / (\|w_i^{t_1}\| \cdot \|w_i^{t_2}\|), \qquad (7)$$

where $w_i^{t_1}$ and $w_i^{t_2}$ are both vectorized to compute their inner products and norms. We set the threshold of selective computation as the average of such difference between model updates at different IoT devices. Since the scale of cosine distance could change during FL training [50], such averaging can add some adaptivity to the threshold.

6 Privacy Protection

Existing work proved that sparsification can reduce the gradient inversion's capability of estimating the training data from a trained model [68], and we showed in Section 5.1 that sparsification causes only negligible impact on error compensation. Hence, in FedDC we will also exploit such sparsification to prevent the IoT device's local data samples from being precisely estimated by gradient inversion.

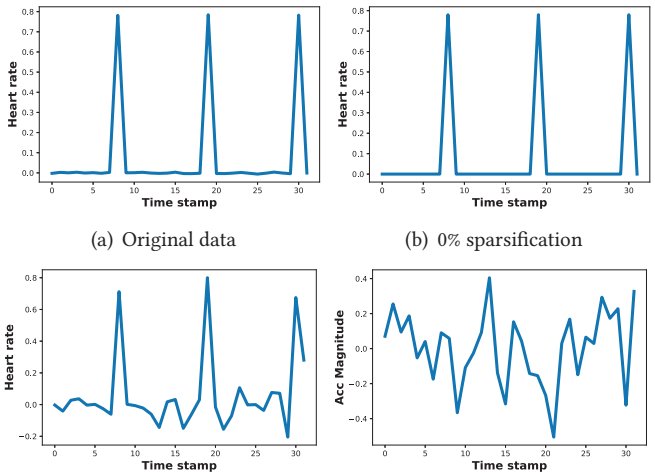
As shown in Figure 7, gradient inversion's capability of data estimation largely depends the size of training data being estimated. To ensure sufficient power of privacy protection, we still focus on the most challenging setting show



(a) Original data (b) 0% sparsification

(c) 40% sparsification (d) 95% sparsification

Figure 13: IMU data estimated with different sparsification rates, on ExtraSensory dataset [52]



(a) Original data (b) 0% sparsification

(c) 40% sparsification (d) 95% sparsification

Figure 14: Heart rate data (normalized) estimated with different sparsification rates, on PAMAP2 dataset [43]

in Figure 7(c) where device's local data contains only one data sample, which is the sole target for gradient inversion to estimate. Figures 13 and 14, then, show the effectiveness of different sparsification rates on reducing gradient inversion's capability of data estimation. A sparsification rate of 40% can produce noticeable disturbances in the estimated data, and when the sparsification rate is 95%, the estimated data looks completely different from the original data samples.

To quantitatively verify such privacy protection, we calculate the similarity between estimated and raw data samples, using different metrics including mean square error (MSE) and dynamic time warping (DTW) distance [46]. As shown in Table 2, with a 95% sparsification rate, the estimated data is more like random noise than the original data.

Metric	Dataset	0%	40%	95%	Noise
MSE ↓	PAMAP2	1e-4	0.08	0.33	0.38
	ExtraSensory	1e-5	0.04	0.34	0.42
DTW ↓	PAMAP2	0.07	0.5	1.03	1.14
	ExtraSensory	0.02	0.18	2.1	2.25

Table 2: The similarity between the estimated data and raw data samples, with different sparsification rates

We also used the PAMAP2 dataset to train a MLP model, which is then used to classify the data samples estimated by gradient inversion. Results in Figure 15 show that using a sparsification rate of 95% could effectively make the model's classification results to be nearly random guesses[2]. Note that, since the PAMAP2 dataset we used is a relatively easy one for the HAR classification task, using a simple neural network classifier can achieve good accuracy as shown by the existing work [16]. Therefore, we consider a simple MLP model as capable of recognizing patterns in the data and sufficient for evaluating our approach's capability of privacy protection.

In addition, our proposed gradient sparsification is also compatible with other privacy protection methods, such as local differential privacy (LDP) [51]. More specifically, when LDP is applied, gradient inversion will not aim to recover the original data distribution of clients, but instead to estimate a data distribution that can produce the noisy gradient. Such recovered distribution is then used to compute the noisy gradient on the current global model.

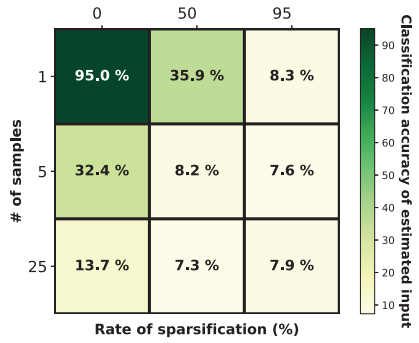


Figure 15: Classification accuracy of estimated data

7 Implementation

As shown in Figure 16, we implement FedDC on 10 IoT devices, including: i) 6 smartphones of different models (Pixel XL, Pixel 2, Pixel 4, Pixel 5, Pixel 7, LG-G5) with different hardware configurations and compute power; ii) 2 Raspberry Pi 4B with a 1.5GHz Cortex-A72 CPU and 4GB memory; and iii) 2 NVidia Jetson Nano with a 128-core Maxwell GPU and

[2]Since the PAMAP2 dataset contains 13 data classes, the classification result is a random guess when the accuracy drops to 7.7%.

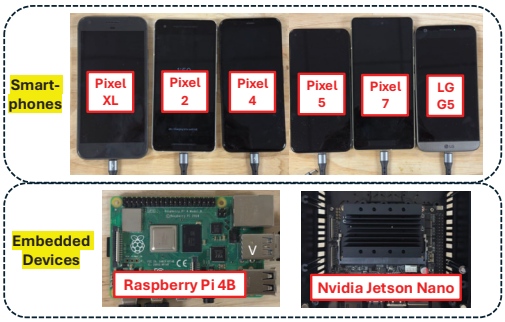


Figure 16: Devices used in experiments

4GB memory. A workstation with a NVidia RTX A5000 GPU is used as server, which communicates with devices via WiFi.

Our implementation builds on the Flower FL framework [6]. Since FedDC retains the IoT devices' local FL training procedure completely unchanged, we adopt the original client FL program provided by Flower, such that each device in every global epoch computes its update using the SGD optimizer with a learning rate of 0.01 and a momentum of 0.5. We modify the aggregation program at the FL server to add the proposed approaches of device delay compensation using gradient inversion, and the server ends a global epoch when 70% of devices have finished uploading their model updates.

To maximize the compute efficiency at FL server and minize the time needed for aggregation, we parallelized the process of gradient inversion on the FL server and the FL training on IoT devices. As shown in Figure 17. the server first aggregates all the model updates from IoT devices that are not affected by device delays and initiates a new global epoch. Then, the server computes gradient inversion for other delayed model updates for the previous epoch, while the IoT devices compute their model updates for the new epoch. Once the gradient inversion at the server completes, the compensated model updates (from the previous epoch) can be aggregated into the global model in the new epoch.

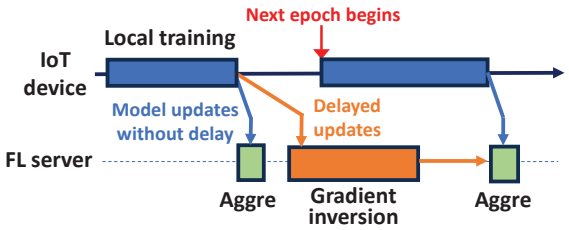


Figure 17: Parallelization of computations

8 Performance Evaluation

We evaluate the performance of FedDC over multiple AIoT datasets in different federated application domains, including human activity recognition (HAR) and hazard site surveillance, in both of which device delays and data heterogeneity are closely correlated.

- **PAMAP2 [43]** with 13 classes of human activities and >2M data samples collected using IMU and heart rate sensors. A 3-layer MLP model is used in FL.
- **ExtraSensory [52]** with over 300k data samples collected using IMU, gyroscope and magnetometer sensors on smartphones. Besides 7 main labels of activities (e.g., standing, laying down, etc), it also provides 109 additional labels describing more specific activity contexts. An 1D-CNN model is used in FL.
- **MDI [38]** with 6,000 images about 6 classes of natural disasters. A pre-trained ResNet18 model is used in FL.

We compare FedDC's performance in classification task with the following baselines that tackle devices delays in FL:

- **Unweighted aggregation (Unweighted)**: Aggregating delayed model updates without applying weights.
- **Weighted aggregation (Weighted)**: Aggregating delayed model updates with weights that are inversely proportional to the amount of device delays [10].
- **FL with asynchronous tiers (Asyn-Tiers)**: It clusters devices into asynchronous tiers based on device delays and uses synchronous FL in each tier [8].
- **First-order compensation (1st-Order)**: It applies Taylor expansion on the gradient of delayed model update and uses its first-order term as estimator [65].
- **Future global weights prediction (W-Pred)**: Assuming delays as pre-known, the future global model is predicted by first-order method above and used to estimate model updates without delays [15].

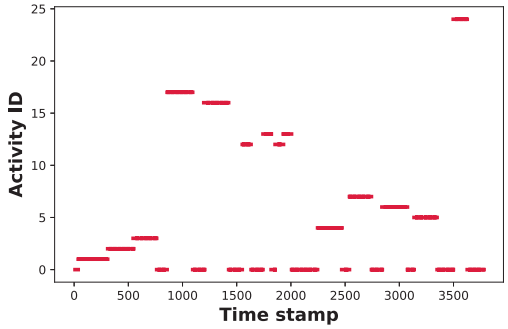
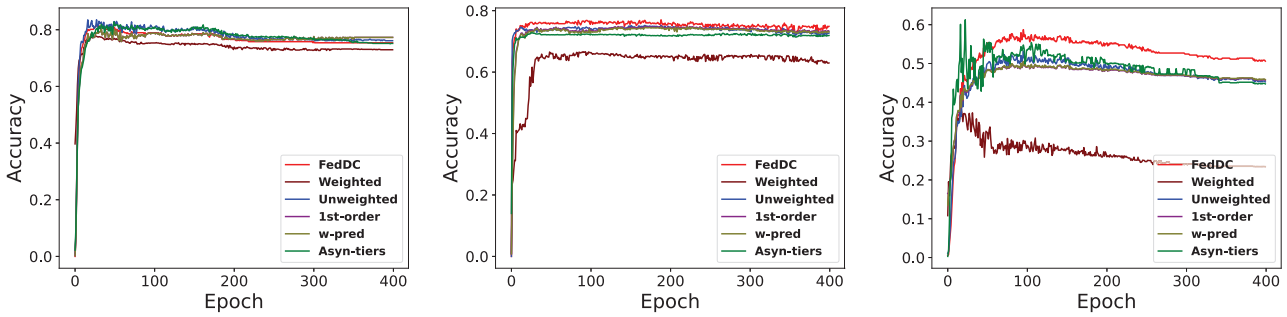


Figure 18: Distribution of data samples' classes over time, from one subject in the PAMAP2 dataset

8.1 Experiment Settings

In all experiments, FedAvg [35] is used for aggregating model updates. Hence, Unweighted aggregation is FedAvg with delays, and Weighted aggregation applies extra weights to model updates in FedAvg 1st-Order, W-pred, and FedDC further modify such weights for compensation, and Asyn-Tiers separately uses FedAvg in each tier. Note that the usage of FedAvg is independent from FedDC and other baselines, and can be replaced by other FL frameworks such as FedProx



(a) Low delay (b) Medium delay (c) High delay

Figure 19: FL training procedure with different amounts of delays on the PAMAP2 dataset

[29]. For Weighted aggregation, we set the weights following [48] as $1/(1+e^{a(\tau-b)})$, where τ is the amount of delays in global epochs and we set a=0.25 and b=10. For Asyn-Tiers, we set 3 asynchronous tiers that are also weighted by the number of devices in different tiers [8].

Data heterogeneity. We enforce different levels of data heterogeneity by partitioning the dataset to IoT devices in different ways. First, in most AIoT datasets, data samples collected from a subject within a short time window will fall into the same class, as exemplified in Figure 18. Hence, being different from existing FL studies which set data heterogeneity using a Dirichlet distribution to sample client datasets [70], we emulate high data heterogeneity using a short time window, during which an IoT device only has data in one class. Similarly, low data heterogeneity is emulated over a long time window, during which data samples in different classes are continuously added to IoT devices.

Device delays. For each dataset, we randomly select three data classes that will be affected by low, medium and high amounts of device delays, respectively. In each FL training epoch, the specific device delays are sampled from a Gamma distribution: for high delays, its mean value is set to 10× of the average time of a global epoch. The mean values for medium and low delays are set as 5× and 2×, respectively.

With the parallelization approach described in Section 7, the duration of a global epoch in AIoT is mainly determined by the IoT devices' local training, which is usually slow due to the devices' limited local computing power. On the devices we used in evaluations, the average durations for local training with the three datasets are 28.1s (PAMAP2), 55.6s (ExtraSensory), and 23.7s (MDI), respectively. Note that, such durations are decided by the weakest client in the FL system.

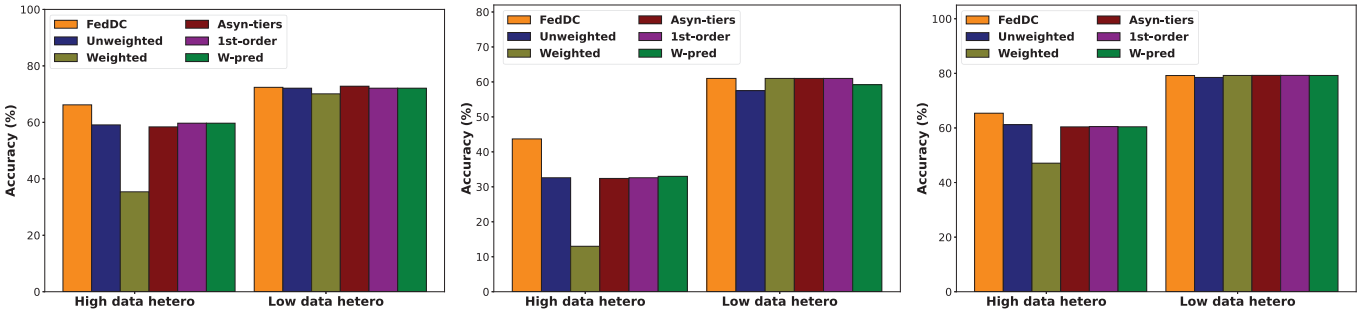
8.2 FL Performance with Different Amounts of Device Delays

We first evaluate the FL model's accuracy in data classes affected by different amounts of device delays. Given that the accuracy across different data classes in a dataset could significantly vary, even for the same method, we report model accuracy on data classes affected by delay as the *relative improvement* compared to unweighted aggregation, to clearly show the performance differences among different methods. Besides, we also report the overall performance on the entire dataset, as the absolute percentage of classification accuracy. Our method only adds extra computations of gradient inversion on the server, and the communication cost between the server and clients remains the same as that in conventional FL. Hence, we do not report the communication cost in the evaluations of this paper.

As shown in Table 3, when the amount of device delays is low, FedDC results in slight degradation in FL performance because the error in delay compensation outweighs the error caused by delayed model updates. However, when the amount of device delays increases, the delay compensation in

	PAMAP2 / MLP				ExtraSensory / 1D-CNN				MDI / 2D-CNN			
Delay	low	medium	high	overall	low	medium	high	overall	low	medium	high	overall
Unweighted	0.0%	0.0%	0.0%	69.9%	0.0%	0.0%	0.0%	48.7%	0.0%	0.0%	0.0%	76.5%
Weighted	-5.4%	-13.9%	-43.5%	68.9%	-18.4%	-46.5%	-62.3%	47.5%	-3.7%	-14.1%	-23.3%	76.3%
Asyn-tiers	+0.7%	+0.4%	-0.5%	69.9%	-2.0%	+0.8%	-2.9%	48.9%	-1.3%	+1.8%	0.0%	76.5%
1st-Order	+2.3%	+1.5%	+0.6%	70.01%	+3.6%	+2.5%	-2.2%	49.3%	-0.5%	+1.7%	+0.7%	76.9%
W-Pred	+2.6%	+1.3%	+0.6%	70.0%	+0.4%	+1.5%	-1.3%	49.6%	-0.2%	+1.7%	+0.4%	76.8%
FedDC	-1.9%	+5.4%	+12.3%	70.6%	-3.0%	+16.9%	+34.2%	50.3%	-1.2%	+4.1%	+7.7%	76.8%

Table 3: The trained model's accuracy in data classes affected by device delays, with different amounts of device delays. Accuracy is shown as the relative improvement compared to unweighted aggregation.



(a) PAMAP2 [43] (b) ExtraSensory [52] (c) MDI [38]

Figure 20: FL performance with different levels of data heterogeneity

FedDC can significantly improve FL performance compared to baselines. In particular, with a high amount of device delay, FedDC outperforms the best baseline (W-Pred) by 35.5% on ExtraSensory dataset and 11.7% on PAMAP2 dataset. Note that the average classification accuracy on ExtraSensory dataset is around 50% and the lowest among three datasets. This means that FedDC can achieve the most performance improvement in challenging federated AIoT scenarios.

Meanwhile, Table 3 showed that the average accuracy over all classes in FedDC remain nearly the same with slight improvement. Since device delays in our experiment settings only affect 3 data classes which constitute only a small portion of the entire dataset as described in Section 8.1, these results show that FedDC's delay compensation did not affect the FL performance in other data classes.

Results in Figure 19 further show that, besides achieving higher accuracy in the trained model, FedDC also improves the quality and stability of training, especially with high amounts of device delays.

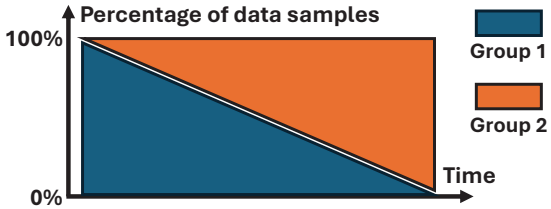


Figure 21: Variant distribution of global data over time

8.3 FL Performance with Different Levels of Data Heterogeneity

We also compared FedDC's performance with baselines when different levels of data heterogeneity are present. Experiment results in Figure 20 show that, when the data heterogeneity is high, weighted aggregation suffers significant performance drop due to the biased model being trained. In contrast, FedDC can effectively improve the FL performance on all the three datasets, and such improvement is similarly more significant on more challenging datasets such as ExtraSensory. On the other hand, when the level of data heterogeneity is low, FedDC retains the similar level of FL performance with other baselines, indicating that our approach to switching back to vanilla FL, as described in Section 4.3, can effectively avoid unnecessary errors in delay compensation.

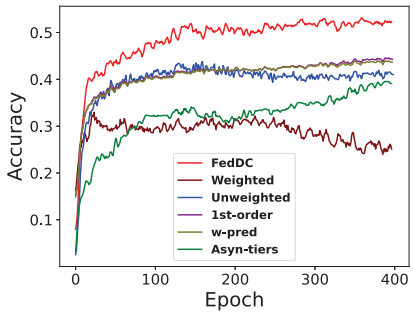
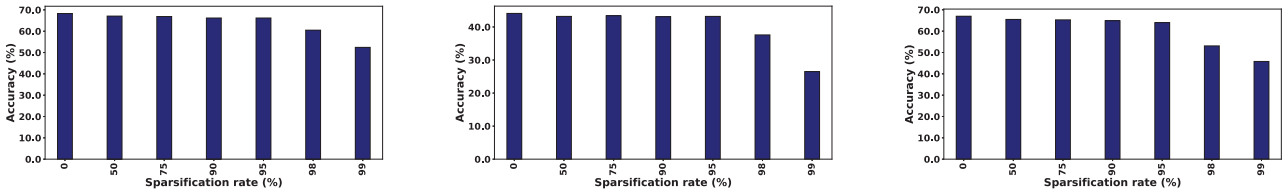


Figure 22: Slow convergence of FL training with the variant global data distribution, on PAMAP2 dataset

8.4 FL Performance under Variant Global Data Distribution

We further consider a more challenging FL scenario where the global data distribution continuously varies over time, resulting in much slower convergence of FL training. Such variance of data distribution commonly exists in practical AIoT applications. For example, in the HAR application, human activities in daytime and nighttime can significantly differ. As shown in Figure 21, we emulate such variant data distribution by dividing the global dataset corpus into two parts, and smoothly transit the current global data from one part to another part as FL training progresses. In practice, such transition is concurrently ongoing on all IoT devices.

Based on such settings, results in Table 4 show that FedDC significantly outperforms all the baselines on different AIoT datasets. Comparing results in Table 4 with those in Table 3 where data on each IoT device is fixed, we found that FedDC achieves larger performance improvement when the global data distribution is variant. This is because with the



(a) PAMAP2 [43] (b) ExtraSensory [52] (c) MDI [38]

Figure 23: FL performance with different rates of gradient sparsification

	PAMAP2	EtraSensory	MDI
Unweighted	38.9%	20.3%	65.1%
Weighted	23.8%	0%	59.0%
Asyn-tiers	37.5%	16.7%	66.8%
1st-Order	42.3%	22.4%	65.1%
W-Pred	42.1%	22.4%	65.1%
FedDC	53.5%	34.7%	69.3%

Table 4: FL performance with variant global data distribution and high device delays

variant global data distribution, the FL training converges very slowly and FedDC's delay compensation hence always results in smaller errors compared to directly aggregating delayed model updates into the global model. Such slow convergence of FL training can be verified in Figure 22.

8.5 Computing Cost at the FL Server

Gradient inversion at the FL server is computationally expensive and we proposed techniques in Section 5 to reduce the server's computing costs. As shown in Table 5, applying 95% gradient sparsification (§5.1) and selective computation (§5.2) can significantly reduce such computing costs to be within the duration of one global epoch, when being tested on a NVidia A5000 GPU. Since the IoT devices' local trainings are computed in parallel with FL server's computation, as shown in Figure 17, such reduction of computing cost ensures that computing gradient inversion at the FL server will not incur extra delays in the end-to-end FL training process[3]. Besides, since FedDC does not incur any extra computations at IoT devices, their local computing and communication costs are exactly the same as those in vanilla FL.

FL model	MLP	1D-CNN	2D-CNN
Full Computation	21.63	15.12	19.66
Selective Computation	2.54	2.23	3.71
Selective Computation + 95% SP	0.91	0.82	0.88

Table 5: Computing delay of gradient inversion at the server, measured as the number of global FL epochs

Meanwhile, we also evaluated the impact of these computationally efficient techniques on FL performance. Figure 23 show that using 95% gradient sparsification only results in <2% of FL performance drop.

8.6 Sensitivity Analysis

Switching to vanilla FL. As demonstrated in Section 4.3, we need to switch back from FedDC to vanilla FL at the late stage of FL training. However, since our approach cannot compute the compensation error in the current global epoch, there is always a latency in switching. To investigate the impact of such latency on FL performance, we checked the FL performance with different choices of switching using the PAMAP2 dataset. Results in Figure 24 show that FL performance with switching is significantly higher than that without switch, but using different choices of switching has little impact on FL performance, indicating that FedDC is not sensitive to the latency in switching.

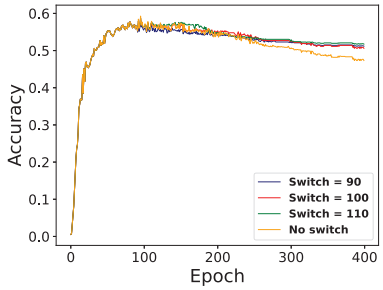


Figure 24: Performance with different switch points

Device's local FL training. FL performance could also be affected by the local training programs being used at IoT devices [42]. To evaluate the performance of FedDC in different FL settings, we use three different optimizers (SGD, SGDW, Adam) for IoT device's local model updates, and results in Figure 25 show that FedDC and all baselines exhibit consistence FL performance with SGD and SGDW optimizers. However, with adaptive optimizers such as Adam that increases the gradient divergence when the devices' local data is not i.i.d. [23], the FL performance significantly drops.

8.7 Scalability Analysis

To further explore the performance of FedDC in large-scale AIoT applications [39], we simulate a larger FL systems with 50 to 300 IoT devices, 10% of which are affected by high

[3]Note that, aggregation of model updates at the FL server is very lightweight and incurs negligible computing overhead in most FL scenarios.

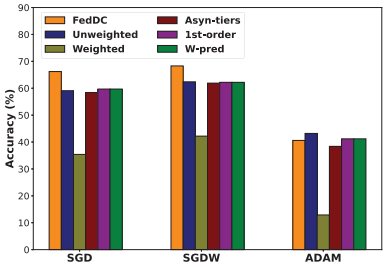


Figure 25: FL performance with different local training optimizers and high amounts of device delays

device delays, using a central FL server. As shown in Figure 26, FedDC can maintain a consistent level of FL performance, even when the number of IoT devices increases by 30×.

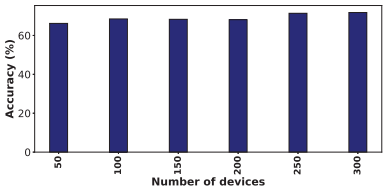


Figure 26: FL performance in large-scale AIoT systems

9 Related Work

Weighted aggregation. Most existing solutions to device delays in FL are based on weighted aggregation [10].These solutions initially introduce metrics to quantify the global model's delay on each client device, which may be based on either time [48, 66] or model discrepancies [54]. Subsequently, these metrics are used to calculate a decay coefficient for slower devices and apply it to model updates. Other approaches adopt more fine-grained weighting [10, 53], where different submodules are weighted individually.

However, weighted aggregation is always biased towards fast devices, and affects global model's accuracy when device delays and data heterogeneity in FL are correlated. Others suggest to use semi-asynchronous FL, where the server either aggregates client model updates at a lower frequency [40] or clusters clients into different asynchronous "tiers" according to their update rates [8]. However, doing so cannot completely eliminate the impact of device delays, because the server's aggregation still involves delayed model updates.

Other data estimation methods. Our approach to compensating device delays using gradient inversion is inspired by the existing work of estimating the device's local training data from the trained model. Some existing work trains a generative model [70] and compels its generated data samples to exhibit high predictive values on the original model. [59] also proposes to directly optimize randomly initialized input data until it performs well on the original model. However, their accuracy of knowledge transfer is generally too low to ensure precise compensation of device delays. Other efforts enhance the quality of knowledge transfer by incorporating natural image priors [33], using another dataset to introduce general knowledge [57], or exploiting the model's activation sparsity [32, 49], but require extra datasets. Moreover, all these methods require the devices' local models to be fully trained, which is usually infeasible in FL.

10 Discussions

Other data modalities in FL. Our evaluations involve data modalities of time series and images, but FedDC can also be applied to other modalities such as text. Since text is decomposed into discrete tokens, we can perform data estimation in the continuous embedding space [68]. Since errors occur when projecting the estimated data from the embedding space into discrete tokens, estimating text data is harder for gradient inversion [12, 14]. This suggests that when applying FedDC to text data, the risk of data privacy leakage is lower.

Handling multimodal data. A common approach to handling multimodal data is to use submodules to extract feature representations from different modalities and then fuse them [5], and this approach also applies to FedDC. Considering that the risk of privacy leakage may vary in each modality [14], different sparsification rates should be used.

More complicated models in FL. The main difficulty of training a complicated neural network model in FedDC is that it incurs much higher computing overhead on IoT devices. Such overhead can be reduced by using a pre-trained model and freezing lower layers. Parameter-efficient training techniques, such as model quantization [55], can also be used. These techniques are orthogonal but applicable to FedDC.

Cost of handling different versions of model. In FedDC, the server needs to tackle multiple versions of models and data, which introduce extra cost in practice. Without loss of generality, we assume that the server is much more powerful than AIoT devices in capabilities of computation, communication and storage. Additionally, the aforementioned parameter-efficient training techniques can also be applied to reduce the computing cost of handling different versions of AI models.

11 Conclusion

In this paper, we present FedDC, a new FL technique that uses gradient inversion in compensating the error caused correlated device delays. Experiment results show that our technique can largely improve the model accuracy while keeping privacy well-protected.

Acknowledgments

We thank the shepherd and reviewers for their comments and feedback. This work was supported in part by National Science Foundation (NSF) under grant number IIS-2205360, CCF-2217003, CCF-2215042, and National Institutes of Health (NIH) under grant number R01HL170368.

References

[1] Lulwa Ahmed, Kashif Ahmad, Naina Said, Basheer Qolomany, Junaid Qadir, and Ala Al-Fuqaha. 2020. Active learning based federated learning for waste and natural disaster image classification. *IEEE Access* 8 (2020), 208518–208531.

[2] Kevin Ashton et al. 2009. That 'internet of things' thing. *RFID journal* 22, 7 (2009), 97–114.

[3] Sheraz Aslam, Michalis P Michaelides, and Herodotos Herodotou. 2020. Internet of ships: A survey on architectures, emerging applications, and challenges. *IEEE Internet of Things journal* 7, 10 (2020), 9714–9727.

[4] Jordi Mongay Batalla, Constandinos X Mavromoustakis, George Mastorakis, Neal Naixue Xiong, and Jozef Wozniak. 2020. Adaptive positioning systems based on multiple wireless interfaces for industrial IoT in harsh manufacturing environments. *IEEE Journal on Selected Areas in Communications* 38, 5 (2020), 899–914.

[5] Khaled Bayoudh, Raja Knani, Fayçal Hamdaoui, and Abdellatif Mtibaa. 2022. A survey on deep multimodal learning for computer vision: advances, trends, applications, and datasets. *The Visual Computer* 38, 8 (2022), 2939–2970.

[6] Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Kwing Hei Li, Titouan Parcollet, Pedro Porto Buarque de Gusmão, et al. 2020. Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390* (2020).

[7] Luca Catarinucci, Danilo De Donno, Luca Mainetti, Luca Palano, Luigi Patrono, Maria Laura Stefanizzi, and Luciano Tarricone. 2015. An IoT-aware architecture for smart healthcare systems. *IEEE internet of things journal* 2, 6 (2015), 515–526.

[8] Zheng Chai, Yujing Chen, Ali Anwar, Liang Zhao, Yue Cheng, and Huzefa Rangwala. 2021. FedAT: A high-performance and communication-efficient federated learning system with asynchronous tiers. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–16.

[9] Ming Chen, Bingcheng Mao, and Tianyi Ma. 2019. Efficient and robust asynchronous federated learning with stragglers. In *International Conference on Learning Representations*.

[10] Yang Chen, Xiaoyan Sun, and Yaochu Jin. 2019. Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation. *IEEE transactions on neural networks and learning systems* 31, 10 (2019), 4229–4238.

[11] Juan Contreras-Castillo, Sherali Zeadally, and Juan Antonio Guerrero-Ibañez. 2017. Internet of vehicles: architecture, protocols, and security. *IEEE internet of things journal* 5, 5 (2017), 3701–3709.

[12] Dimitar I Dimitrov, Mislav Balunović, Nikola Jovanović, and Martin Vechev. 2022. Lamp: Extracting text from gradients with language model priors. *arXiv e-prints* (2022), arXiv–2202.

[13] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. 2020. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in neural information processing systems* 33 (2020), 16937–16947.

[14] Samyak Gupta, Yangsibo Huang, Zexuan Zhong, Tianyu Gao, Kai Li, and Danqi Chen. 2022. Recovering private text in federated learning of language models. *Advances in neural information processing systems* 35 (2022), 8130–8143.

[15] Ido Hakimi, Saar Barkai, Moshe Gabel, and Assaf Schuster. 2019. Taming momentum in a distributed asynchronous environment. *arXiv preprint arXiv:1907.11612* (2019).

[16] Chaolei Han, Lei Zhang, Yin Tang, Wenbo Huang, Fuhong Min, and Jun He. 2022. Human activity recognition using wearable sensors by heterogeneous convolutional neural networks. *Expert Systems with Applications* 198 (2022), 116764.

[17] Anbu Huang, Yuanyuan Chen, Yang Liu, Tianjian Chen, and Qiang Yang. 2020. RPN: A residual pooling network for efficient federated learning. In *ECAI 2020*. IOS Press, 1223–1229.

[18] Kai Huang, Boyuan Yang, and Wei Gao. 2023. Elastictrainer: Speeding up on-device training with runtime elastic tensor selection. In *Proceedings of the 21st Annual International Conference on Mobile Systems, Applications and Services*. 56–69.

[19] Kai Huang, Hanyun Yin, Heng Huang, and Wei Gao. 2024. Towards Green AI in Fine-tuning Large Language Models via Adaptive Backpropagation. *ICLR* (2024).

[20] Kai Huang, Xiangyu Yin, Tao Gu, and Wei Gao. 2024. Perceptual-Centric Image Super-Resolution using Heterogeneous Processors on Mobile Devices. In *ACM MobiCom*.

[21] Kai Huang, Xiangyu Yin, Heng Huang, and Wei Gao. 2025. Modality Plug-and-Play: Runtime Modality Adaptation in LLM-Driven Autonomous Mobile Systems. In *ACM MobiCom*.

[22] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2021. Advances and open problems in federated learning. *Foundations and trends® in machine learning* 14, 1–2 (2021), 1–210.

[23] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. 2020. Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning*. PMLR, 5132–5143.

[24] Latif U Khan, Walid Saad, Zhu Han, Ekram Hossain, and Choong Seon Hong. 2021. Federated learning for internet of things: Recent advances, taxonomy, and open challenges. *IEEE Communications Surveys & Tutorials* 23, 3 (2021), 1759–1799.

[25] Jakub Konečnỳ, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. 2016. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527* (2016).

[26] Hugo Landaluce, Laura Arjona, Asier Perallos, Francisco Falcone, Ignacio Angulo, and Florian Muralter. 2020. A review of IoT sensing applications and challenges using RFID and wireless sensor networks. *Sensors* 20, 9 (2020), 2495.

[27] Chenglin Li, Di Niu, Bei Jiang, Xiao Zuo, and Jianming Yang. 2021. Meta-har: Federated representation learning for human activity recognition. In *Proceedings of the web conference 2021*. 912–922.

[28] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. 2018. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems* 31 (2018).

[29] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems* 2 (2020), 429–450.

[30] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. 2019. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189* (2019).

[31] Zhaohua Li, Le Wang, Guangyao Chen, Muhammad Shafq, et al. 2023. A survey of image gradient inversion against federated learning. *Authorea Preprints* (2023).

[32] Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, et al. 2023. Deja vu: Contextual sparsity for efficient llms at inference time. In *International Conference on Machine Learning*. PMLR, 22137–22176.

[33] Liangchen Luo, Mark Sandler, Zi Lin, Andrey Zhmoginov, and Andrew Howard. 2020. Large-scale generative data-free distillation. *arXiv preprint arXiv:2012.05578* (2020).

[34] Yuzhu Mao, Zihao Zhao, Meilin Yang, Le Liang, Yang Liu, Wenbo Ding, Tian Lan, and Xiao-Ping Zhang. 2023. Safari: Sparsity-enabled

federated learning with limited and unreliable communications. *IEEE Transactions on Mobile Computing* (2023).

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

[36] Elena Mocanu, Decebal Constantin Mocanu, Phuong H Nguyen, Antonio Liotta, Michael E Webber, Madeleine Gibescu, and Johannes G Slootweg. 2018. On-line building energy optimization using deep reinforcement learning. *IEEE transactions on smart grid* 10, 4 (2018), 3698–3708.

[37] Mehdi Mohammadi and Ala Al-Fuqaha. 2018. Enabling cognitive smart cities using big data and machine learning: Approaches and challenges. *IEEE Communications Magazine* 56, 2 (2018), 94–101.

[38] Hussein Mouzannar, Yara Rizk, and Mariette Awad. 2018. Damage Identification in Social Media Posts using Multimodal Deep Learning.. In *ISCRAM*. Rochester, NY, USA.

[39] Dinh C Nguyen, Ming Ding, Pubudu N Pathirana, Aruna Seneviratne, Jun Li, and H Vincent Poor. 2021. Federated learning for internet of things: A comprehensive survey. *IEEE Communications Surveys & Tutorials* 23, 3 (2021), 1622–1658.

[40] John Nguyen, Kshitiz Malik, Hongyuan Zhan, Ashkan Yousefpour, Mike Rabbat, Mani Malek, and Dzmitry Huba. 2022. Federated learning with buffered asynchronous aggregation. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 3581–3607.

[41] Amir M Rahmani, Tuan Nguyen Gia, Behailu Negash, Arman Anzanpour, Iman Azimi, Mingzhe Jiang, and Pasi Liljeberg. 2018. Exploiting smart e-Health gateways at the edge of healthcare Internet-of-Things: A fog computing approach. *Future Generation Computer Systems* 78 (2018), 641–658.

[42] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and H Brendan McMahan. 2020. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295* (2020).

[43] Attila Reiss and Didier Stricker. 2012. Introducing a new benchmarked dataset for activity monitoring. In *2012 16th international symposium on wearable computers*. IEEE, 108–109.

[44] Yichen Ruan, Xiaoxi Zhang, Shu-Che Liang, and Carlee Joe-Wong. 2021. Towards flexible device participation in federated learning. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 3403–3411.

[45] Aaqib Saeed, Flora D Salim, Tanir Ozcelebi, and Johan Lukkien. 2020. Federated self-supervised learning of multisensor representations for embedded intelligence. *IEEE Internet of Things Journal* 8, 2 (2020), 1030–1040.

[46] Hiroaki Sakoe and Seibi Chiba. 1978. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing* 26, 1 (1978), 43–49.

[47] Deepti Sehrawat and Nasib Singh Gill. 2019. Smart sensors: Analysis of different types of IoT sensors. In *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*. IEEE, 523–528.

[48] Guomei Shi, Li Li, Jun Wang, Wenyan Chen, Kejiang Ye, and ChengZhong Xu. 2020. HySync: Hybrid federated learning with effective synchronization. In *2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. IEEE, 628–633.

[49] Jifeng Song, Kai Huang, Xiangyu Yin, Boyuan Yang, and Wei Gao. 2024. Achieving Sparse Activation in Small Language Models. *arXiv preprint arXiv:2406.06562* (2024).

[50] Pu Tian, Weixian Liao, Wei Yu, and Erik Blasch. 2022. WSCC: A weight-similarity-based client clustering approach for non-IID federated learning. *IEEE Internet of Things Journal* 9, 20 (2022), 20243–20256.

[51] Stacey Truex, Ling Liu, Ka-Ho Chow, Mehmet Emre Gursoy, and Wenqi Wei. 2020. LDP-Fed: Federated learning with local differential privacy. In *Proceedings of the third ACM international workshop on edge systems, analytics and networking*. 61–66.

[52] Yonatan Vaizman, Katherine Ellis, and Gert Lanckriet. 2017. Recognizing detailed human context in the wild from smartphones and smartwatches. *IEEE pervasive computing* 16, 4 (2017), 62–74.

[53] Qizhao Wang, Qing Li, Kai Wang, Hong Wang, and Peng Zeng. 2021. Efficient federated learning for fault diagnosis in industrial cloud-edge computing. *Computing* 103, 10 (2021), 2319–2337.

[54] Qiyuan Wang, Qianqian Yang, Shibo He, Zhiguo Shi, and Jiming Chen. 2022. Asyncfeded: Asynchronous federated learning with euclidean distance based adaptive weight aggregation. *arXiv preprint arXiv:2205.13797* (2022).

[55] Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, and Jian Cheng. 2016. Quantized convolutional neural networks for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4820–4828.

[56] Cong Xie, Sanmi Koyejo, and Indranil Gupta. 2019. Asynchronous federated optimization. *arXiv preprint arXiv:1903.03934* (2019).

[57] Ziqi Yang, Jiyi Zhang, Ee-Chien Chang, and Zhenkai Liang. 2019. Neural network inversion in adversarial setting via background knowledge alignment. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 225–240.

[58] Hongxu Yin, Arun Mallya, Arash Vahdat, Jose M Alvarez, Jan Kautz, and Pavlo Molchanov. 2021. See through gradients: Image batch recovery via gradinversion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 16337–16346.

[59] Hongxu Yin, Pavlo Molchanov, Jose M Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K Jha, and Jan Kautz. 2020. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 8715–8724.

[60] Xiangyu Yin, Kai Huang, Erick Forno, Wei Chen, Heng Huang, and Wei Gao. 2023. PTEase: Objective Airway Examination for Pulmonary Telemedicine using Commodity Smartphones. In *Proceedings of the 21st Annual International Conference on Mobile Systems, Applications and Services*. 110–123.

[61] Jing Zhang and Dacheng Tao. 2020. Empowering things with intelligence: a survey of the progress, challenges, and opportunities in artificial intelligence of things. *IEEE Internet of Things Journal* 8, 10 (2020), 7789–7817.

[62] Rui Zhang, Song Guo, Junxiao Wang, Xin Xie, and Dacheng Tao. 2022. A survey on gradient inversion: Attacks, defenses and future directions. *arXiv preprint arXiv:2206.07284* (2022).

[63] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. 2020. idlg: Improved deep leakage from gradients. *arXiv preprint arXiv:2001.02610* (2020).

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

[65] Shuxin Zheng, Qi Meng, Taifeng Wang, Wei Chen, Nenghai Yu, Zhi-Ming Ma, and Tie-Yan Liu. 2017. Asynchronous stochastic gradient descent with delay compensation. In *International conference on machine learning*. PMLR, 4120–4129.

[66] Chendi Zhou, Hao Tian, Hong Zhang, Jin Zhang, Mianxiong Dong, and Juncheng Jia. 2021. TEA-fed: time-efficient asynchronous federated learning for edge computing. In *Proceedings of the 18th ACM International Conference on Computing Frontiers*. 30–37.

[67] Yuhao Zhou, Qing Ye, and Jiancheng Lv. 2021. Communication-efficient federated learning with compensated overlap-fedavg. *IEEE Transactions on Parallel and Distributed Systems* 33, 1 (2021), 192–205.

[68] Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep leakage from gradients. *Advances in neural information processing systems* 32 (2019).

[69] Xudong Zhu, Hui Li, and Yang Yu. 2019. Blockchain-based privacy preserving deep learning. In *Information Security and Cryptology: 14th International Conference, Inscrypt 2018, Fuzhou, China, December 14-17, 2018, Revised Selected Papers 14*. Springer, 370–383.

[70] Zhuangdi Zhu, Junyuan Hong, and Jiayu Zhou. 2021. Data-free knowledge distillation for heterogeneous federated learning. In *International conference on machine learning*. PMLR, 12878–12889.