Tackling Intertwined Data and Device Heterogeneities in Federated Learning with Unlimited Staleness

Haoming Wang and Wei Gao

University of Pittsburgh
hw.wang@pitt.edu,weigao@pitt.edu

Abstract

Federated Learning (FL) can be affected by data and device heterogeneities, caused by clients' different local data distributions and latencies in uploading model updates (i.e., staleness). Traditional schemes consider these heterogeneities as two separate and independent aspects, but this assumption is unrealistic in practical FL scenarios where these heterogeneities are intertwined. In these cases, traditional FL schemes are ineffective, and a better approach is to convert a stale model update into a unstale one. In this paper, we present a new FL framework that ensures the accuracy and computational efficiency of this conversion, hence effectively tackling the intertwined heterogeneities that may cause unlimited staleness in model updates. Our basic idea is to estimate the distributions of clients' local training data from their uploaded stale model updates, and use these estimations to compute unstale client model updates. In this way, our approach does not require any auxiliary dataset nor the clients' local models to be fully trained, and does not incur any additional computation or communication overhead at client devices. We compared our approach with the existing FL strategies on mainstream datasets and models, and showed that our approach can improve the accuracy by up to 25% and reduce the number of required training epochs by up to 35%.

Code — https://github.com/pittisl/intertwined-FL
Extended version — https://arxiv.org/abs/2309.13536

Introduction

Federated Learning (FL) (McMahan 2016) could be affected by both data and device heterogeneities. *Data heterogeneity* is the heterogeneity of non-i.i.d. data distributions on different clients, which make the aggregated global model biased and reduces model accuracy (Konečný 2016; Zhao 2018). *Device heterogeneity* arises from clients' variant latencies in uploading their local model updates to the server, due to their different local resource conditions (e.g., computing power, network link speed, etc). An intuitive solution to device heterogeneity is asynchronous FL, which does not wait for slow clients but updates the global model whenever having received a client update (Xie and Gupta. 2019). In this case, if a slow client's excessive latency is longer than a training epoch, it will use an outdated global model to compute its model update, which will be *stale* when aggregated at the server and affect model accuracy. To tackle *staleness*, weighted aggregation can be used to apply reduced weights on stale model updates (Chen and Jin. 2019; Wang 2022).

Most existing work considers data and device heterogeneities as two separate and independent aspects in FL (Zhou 2021). This assumption, however, is unrealistic in many FL scenarios where these two heterogeneities are *intertwined*: data in certain classes or with particular features may only be available at some slow clients. For example, in FL for hazard rescue (Ahmed et al. 2020), only devices at hazard sites have crucial data about hazards, but they usually have limited connectivity or computing power to timely upload model updates. Similar situations could also happen in FL scenarios where data with high importance to model accuracy is scarce and hard to obtain, such as disease evaluation in smart health, where only few patients have severe symptoms but are very likely to report symptoms with long delays due to their worsening conditions (Chen et al. 2017).

In these cases, if reduced weights are applied to stale model updates from slow clients, important knowledge in these updates will not be sufficiently learned and hence affects model accuracy. Instead, a better approach is to equally aggregate all model updates and convert a stale model update into a unstale one, but existing techniques for such conversion are limited to a small amount of staleness. For example, first-order compensation can be applied on the gradient delay (Zheng et al. 2017), by assuming staleness in FL is smaller than one epoch to ignore all the high-order terms in the difference between stale and unstale model updates (Zhou and Lv. 2021). However, in the aforementioned FL scenarios, it is common to witness excessive or even unlimited staleness, and our experiments in show that the compensation error will quickly increase with staleness.

To efficiently tackle the intertwined data and device heterogeneities with unlimited staleness, in this paper we present a new FL framework that uses gradient inversion at the server to convert stale model updates, by mimicking the local models' gradients produced with their original training data (Zhu and Han. 2019a). The server inversely computes the gradients from clients' stale model updates to obtain an estimated distribution of clients' training data, such that a model trained with the estimated data distribution will ex-

hibit a similar loss surface as that of using clients' original training data. The server uses such estimated data distributions to retrain the current global model, as estimations of clients' unstale model updates. Compared to other model conversion methods, such as training an extra generative model (Yang 2019) or optimizing input data with constraints (Yin 2020), our approach has the following advantages:

- Our approach retains the clients' FL procedure to be unchanged, and hence does not incur any additional computation or communication overhead at client devices, which usually have weak capabilities in FL scenarios.
- Our approach does not require any auxiliary dataset nor the clients' local models to be fully trained, and can hence be widely applied to practical FL scenarios.
- In our approach, the server will not recover any original samples or labels of clients' local training data, and hence avoids impairing the clients' data privacy.

We evaluated our proposed technique by comparing with the mainstream FL schemes on multiple datasets and models. Experiment results show that when tackling intertwined data and device heterogeneities with unlimited staleness, our technique can significantly improve the trained model accuracy by up to 25% and reduce the required amount of training epochs by up to 35%. Since clients in FL need to compute and upload model updates to the server in every training epoch, such reduction of training epochs largely reduces the computing and communication overhead at clients.

More technical details about our approach and experiment results can be found in our extended version on ArXiv: https://arxiv.org/abs/2309.13536. In the rest of this paper, we will refer to different sections of the Technical Appendix, which can also be found in this extended version.

Background and Motivation

In this section, we present preliminary results that demonstrate the ineffectiveness of existing methods in tackling intertwined data and device heterogeneities, hence motivating our proposed approach using gradient inversion.

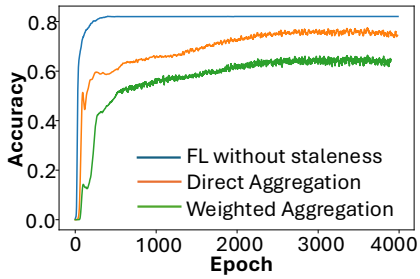


Figure 1: The impact of staleness in FL

Tackling Intertwined Heterogeneities in FL

Weighted aggregation is the most common method to address staleness in FL (Chen and Jin. 2019; Wang 2022), but brings improper bias towards fast clients and misses important knowledge in slow clients' model updates, when data and device heterogeneities are intertwined. To show this, we conducted experiments using a real-world disaster image dataset (Mouzannar, Rizk, and Awad 2018), which contains 6k images of 5 disaster classes (e.g., fires and floods) with different levels of damage severity. In FL of 100 clients, we set *data heterogeneity* as that each client only contain samples in one data class, and set *device heterogeneity* as a staleness of 100 epochs on 15 clients with images of severe damage. When using this dataset to fine-tune a pre-trained ResNet18 model, results in Figure 1 show that staleness leads to large degradation of model accuracy, and weighted aggregation results in even lower accuracy than direct aggregation, because contributions from images of severe damage on stale clients are reduced by the weights[1].

On the other hand, if we increase the contributions from stale clients by using larger weights, although the model accuracy on these images of severe damage will improve, the larger weights will amplify the impacts of errors contained in stale model updates and hence affect the model's overall accuracy in other data classes. Detailed results can be found in Appendix B.

In practical scenarios such as natural disasters, such large or unlimited staleness is common due to interruptions in communication at disaster sites, and the staleness is too large for server to wait for any slow clients. The large performance degradation of weighted aggregation, then, motivates us to instead convert stale model updates to unstale ones.

The only existing work on such conversion, to our best knowledge, uses the first-order Taylor expansion to compensate for errors in stale model updates (Zheng et al. 2017). For a stale update $g(w_{t-\tau})$, the estimated unstale update is:

$$g(w_t) \approx g(w_{t-\tau}) + \nabla g(w_{t-\tau})(w_t - w_{t-\tau}). \quad (1)$$

Since the Hessian matrix $\nabla g(w_{t-\tau})$ is difficult to compute for neural networks, it is approximated as

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

where λ is an empirical hyper parameter. However, this method can only applies to small amounts of staleness (Zhou, Ye, and Lv 2021; Li et al. 2023a; Tian et al. 2021), in which the high-order terms in the Taylor expansion can be negligible. To verify this, we use the same experiment setting as above and vary the amount of staleness from 0 to 50 epochs. As shown in Table 1, the error caused by high-order terms in Taylor expansion, measured in cosine distance and L1-norm difference with the unstale model updates, both significantly increase when staleness increases. These results motivate us to design techniques that ensure accurate conversion with unlimited staleness.

Gradient Inversion

Our proposed approach builds on the existing techniques of gradient inversion (Zhu and Han. 2019b), which recovers the original training data from the gradient of a trained model. Its basic idea is to minimize the difference between

[1]In synchronous FL, stale updates will be simply skipped, corresponding to applying zero weights on these updates. Hence, similar performance degradation is also expected for synchronous FL.

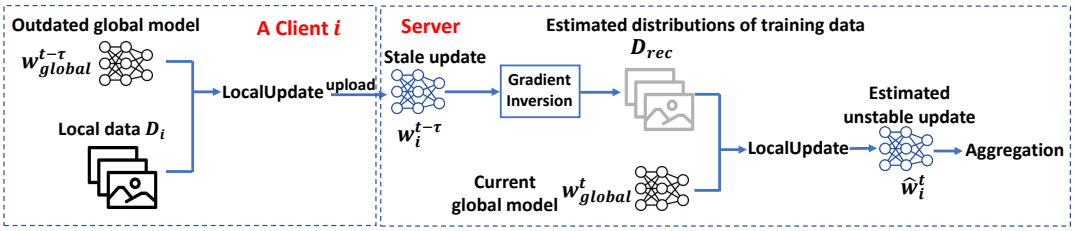


Figure 2: Our proposed method of tackling intertwined data and device heterogeneities in FL

Staleness (epoch)	5	10	20	50
Cos-dist error	0.08	0.22	0.33	0.53
L1-norm error	0.009	0.018	0.31	0.052

Table 1: Errors caused by high-order terms in Taylor expansion when using (Zheng et al. 2017)

the trained model's gradient and the gradient computed from the recovered data. Denote a batch of training data as (x, y) where x denotes input data and y denotes labels, gradient inversion 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', y') is the recovered data, w^{t-1} is the trained model, $L[\cdot]$ is model's loss function, and g^t is the gradient calculated with training data and w^{t-1}. This problem can be solved using gradient descent to iteratively update (x', y').

The quality of recovered data relates to the amount of data samples recovered. Recovering a larger dataset will confuse the learned knowledge across different data samples and reduce the quality of recovered data, and existing methods are limited to recovering a small batch (<48) of data samples (Yin 2021; Geiping 2020; Zhao and Bilen. 2020). This limitation, however, contradicts with the typical size of clients' datasets in FL, which are usually more than hundreds of samples (Wu et al. 2023; Reddi et al. 2020). This limitation indicates that we may utilize gradient inversion to estimate clients' training data distributions without revealing individual samples of clients' local data.

Method

In this paper, we consider a *semi-asynchronous* FL scenario where some normal clients follow synchronous FL and some slow clients update asynchronously (Chai 2021). In this case, we measure staleness by the number of epochs that slow clients' updates are delayed. At time t[2], a normal client i provides its model update as

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

where $LocalUpdate[\cdot]$ is client i's local training program, which uses the current global model w_{global}^t and client i's local dataset D_i to produce w_i^t. When the client i's model update is delayed, the server will receive a stale model update from i at time t as

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

where the amount of staleness is indicated by τ and $w_i^{t-\tau}$ is computed from an outdated global model $w_{global}^{t-\tau}$.

Due to intertwined data and device heterogeneities, we consider that the received $w_i^{t-\tau}$ contains unique knowledge about D_i that is only available from client i, and such knowledge should be sufficiently incorporated into the global model. To do so, as shown in Figure 2, the server uses gradient inversion described in Eq. (3) to recover an intermediate dataset D_{rec} from $w_i^{t-\tau}$. Being different from the existing work of gradient inversion (Zhu and Han. 2019b) that aims to fully recover the client i's training data D_i, we only expect D_{rec} to represent the similar data distribution with D_i.

The server then computes an estimation of w_i^t from $w_i^{t-\tau}$, namely $\hat{w}_i^t$, by using D_{rec} to train its current global model w_{global}^t, and aggregates $\hat{w}_i^t$ with model updates from other clients to update its global model in the current epoch. During this procedure, the server only receives the stale model update $w_i^{t-\tau}$ from client i, which does not expose i's local data D_i to the server. The client i does not need to perform any extra computations for such estimation of $\hat{w}_i^t$, either.

Estimating Local Data Distributions from Stale Model Updates

To compute D_{rec}, we first fix the size of D_{rec} and randomly initialize each data sample and label in D_{rec}. Then, we iteratively update D_{rec} by minimizing

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

using gradient descent, where $Disparity[\cdot]$ is a metric to evaluate how much $w_i^{t-\tau}$ changes if being retrained using D_{rec}. In FL, a client's model update comprises multiple local training steps instead of a single gradient. Hence, to use gradient inversion in FL, we substitute the single gradient computed from D_{rec} in Eq. (3) with the local training outcome using D_{rec}. In this way, since the loss surface in the model's weight space computed using D_{rec} is similar to that using D_i, we can expect a similar gradient being computed.

We first visualize it by using MNIST dataset to train LeNet model. Figure 3 shows that, the loss surface computed using D_{rec} is similar to that using D_i in the proximity of $(w_{global}^{t-\tau})$, and the computed gradient is very similar, too.

[2]In the rest of this paper, without loss of generality, we use the notation of time t to indicate the t-th epoch in FL training.

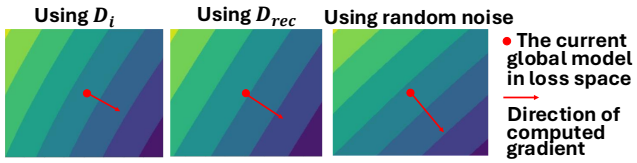
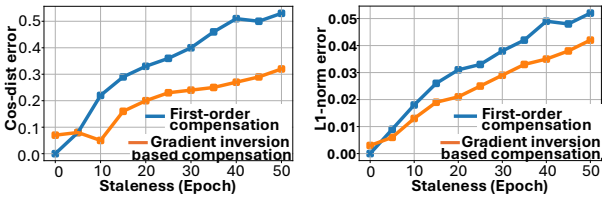


Figure 3: Visualization the loss surface and gradient computed using D_{rec}, D_i, and random noise data

To verify the accuracy of using $\hat{w}_i^t$ to estimate w_i^t, we compare this estimation with first-order estimation, by computing their discrepancies with the true unstale model update under different amounts of staleness. Results in Figure 4 show that, compared to First-order Compensation(Zheng et al. 2017), our estimation based on gradient inversion can reduce the estimation error by up to 50%, especially when staleness excessively increases to more than 50 epochs.

Another key issue is how to decide the size of D_{rec}. Since gradient inversion is equivalent to data resampling in the original training data's distribution, a sufficiently large size of D_{rec} is necessary to ensure unbiased data sampling and sufficient minimization of gradient loss through iterations. On the other hand, when the size of D_{rec} is too large, the computational overhead of each iteration would be unnecessarily too high. More details about how to decide the size of D_{rec} are in Appendix D. Further results about our method's error with various local training programs can also be found in Appendix E.



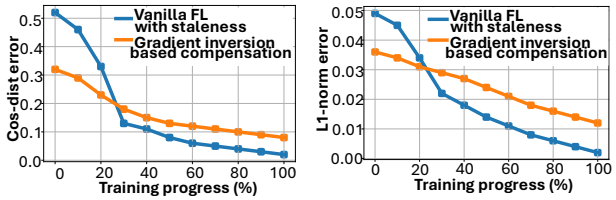
(a) Cosine distance (b) L1-norm difference

Figure 4: Our method of gradient inversion based estimation has smaller error compared to that of first-order estimation

Switching back to Vanilla FL in Later Stages of FL

As shown in Figure 4, the estimation made by gradient inversion also contains errors, because the gradient inversion loss can not be reduced to zero. As the FL training progresses and the global model converges, the difference between the previous and current global models will reduce to 0, and hence the difference between stale and unstale model updates will also reduce, eventually to 0. In this case, in the late stage of FL training, the error in our estimated model update ($\hat{w}_i^t$) will exceed that of the original stale model update $w_i^{t-\tau}$.

To verify this, we conducted experiments by training the LeNet model with the MNIST dataset, and evaluated the average values of $E_1(t) = Disparity[\hat{w}_i^t; w_i^t]$ and $E_2(t) = Disparity[w_i^{t-\tau}; w_i^t]$ across different clients, using both cosine distance and L1-norm difference as the metric. Results in Figure 5 show that at the final stage of FL training, $E_2(t)$ is always larger than $E_1(t)$.



(a) Cosine Distance (b) L1-norm difference

Figure 5: Comparison of model updates' estimation error as the FL training progresses

Deciding the switching point. Hence, in the late stage of FL training, it is necessary to switch back to vanilla FL and directly use stale model updates in aggregation. The difficulty of deciding such switching point is that the true unstale model update (w_i^t) is unknown at time t. Instead, the server will be likely to receive w_i^t at a later time, namely $t + \tau'$. Therefore, if we found that $E_1(t) > E_2(t)$ at time $t + \tau'$ when the server receives w_i^t at $t + \tau'$, we can use $t + \tau'$ as the switching point instead of t. Doing so will result in a delay in switching, but our experiment results in Table 2 with different switching points show that the FL training is insensitive to such delay.

Switch point (epoch)	None	135	155	175
Model accuracy	59.3%	68.1%	67.4%	67.5%

Table 2: FL training results with different switching points. $E_1(t) > E_2(t)$ when t=155, but different switching points exhibit very similar training performance.

In practice, when we make such switch, the model accuracy in training will experience a sudden drop due to the inconsistency of gradients between $\hat{w}_i^t$ and $w_i^{t-\tau}$. To avoid such sudden drop, at time $t + \tau'$, instead of immediately switching to using $\hat{w}_i^t$ in server's model aggregation, we use a weighted average of $\gamma\hat{w}_i^t + (1-\gamma)w_i^{t-\tau}$ in aggregation, so as to ensure smooth switching. γ linearly decays from 1 to 0 within a time window, and the length of this window can be flexibly adjusted to accommodate the optimization of model accuracy. Experiment results in Table 3 show that, when this length is set to 10% of training time before reaching the switching point, the model accuracy is maximized.

Time of decay	0%	5%	10%	20%
Model accuracy	67.4%	69.0%	70.2%	69.8%

Table 3: The time needed for γ to decay from 1 to 0

Computationally Efficient Gradient Inversion

Our basic design rationale is to retain the clients' FL procedure to be unchanged, and offload all the extra computations incurred by gradient inversion to the server. In this way, we

can then focus on reducing the server's computing cost of gradient inversion, which is caused by the large amount of iterations involved, using the following two methods.

First, we reduce complexity of the objective function in gradient inversion by sparsification, which only involve the important gradients with large magnitude into iterations of gradient inversion. Existing work has verified that gradients in mainstream models are highly sparse and only few gradients have large magnitudes (Lin et al. 2017). Hence, we use a binary mask $Mask[\cdot]$ to selecting elements in $w_i^{t-\tau}$ with the top-K magnitudes and only involve these elements to gradient inversion. As shown in Table 4, by only involving the top 5% of gradients, we can reduce around 80% of computation measured as the number of iterations in gradient inversion, with very slight increase in the error of estimating unstale model updates. Besides, we further explored the impact of such error caused by sparsification on the model accuracy, and results are in Appendix F.

Sparsification rate	0%	90%	95%	99%
Reduction of comput. (%)	0%	68%	80%	93%
Estimation error	0.28	0.29	0.31	0.57

Table 4: Reduction of computation and error of estimating unstale model updates, with different sparsification rates

Since in most cases the clients' local data remains fixed, we do not need to start iterations of gradient inversion every time from a random initialization, but could instead optimize D_{rec} from those calculated in the previous training epochs. Our experiments in Table 5 show that, when the clients' local data remains fixed, we can further reduce the amount of iterations in gradient inversion by another 43%. Even if such client data is only partially fixed (e.g., changed by 20%), we can still achieve non-negligible reduction of such iterations.

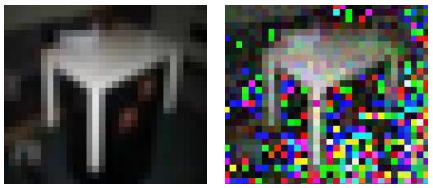
Amount of data changed	0%	5%	20%	50%
Computation saved	43%	21%	12%	6%

Table 5: The number of iterations in gradient inversion with different percentages of changes in clients' local data

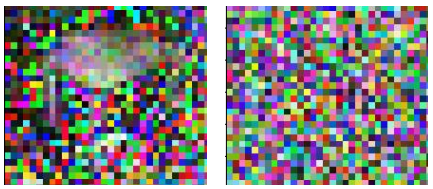
Note that, we only apply gradient inversion to stale model updates containing unique knowledge not present in other model updates. Besides, most FL systems (Charles et al. 2021) keep the number of clients in a global round constant. Once such number is sufficient (e.g., 10-50 even for FL with thousands of clients), further increasing such number yields little performance gains but increases overhead and causes catastrophic training failure (Ro et al. 2022). Hence, the server's overhead of gradient inversion, even in large-scale FL systems, will not largely increase. Such scalability is further discussed in Appendix G.

Protecting Clients' Data Privacy

Although we used gradient inversion to estimate local data distributions from stale model updates, in most FL settings, it would be difficult or nearly impossible for the server to recover, either the stale clients' local data or the labels, from the knowledge about such distributions, especially when applying the sparsification method described before.



(a) Original image (b) 30% sparsification



(c) 75% sparsification (d) 95% sparsification

Figure 6: Recovered images under different sparsification rates, with the CIFAR-10 dataset and the LeNet model

Protecting data samples. The difficulty of recovering clients' local data samples is proportional to the size of clients' local data and the complexity of local training. However, even under the easiest scenario where client's dataset only contains one sample and local training is just one-step gradient descent, such recovery will still be unsuccessful.

More specifically, although gradient inversion can recover the majority of data samples' pixels as shown in Figure 6(a) when no sparsification is applied, the quality of such recovery quickly drops when moderate sparsification is applied, as shown in Figure 6(c) and 6(d). This is because sparsification effectively reduces the scope of knowledge available for gradient inversion to recover data. Results in Table 6 with multiple perceptual image quality metrics, including LPIPS (Zhang 2018) and FID (Heusel et al. 2017), further verify that such recovered images cannot be recognized in human eyes. Essentially, when 95% sparsification rate is applied, the quality of recovered images is similar to that of noise. We also assessed the possibility of a neural network classifier (e.g., a ResNet-18 model) to recognize the recovered images. Results in the last row of Table 6 show that with the 95% sparsification rate, the classification accuracy is nearly equivalent to random guessing.

Besides, since our method only modifies the FL operations on the server and keeps other FL steps (e.g., the clients' local model updates and client-server communication) unchanged, statistical privacy methods, such as differential privacy, can also be applied to local clients in our approach, just like how it applies to vanilla FL. Each client can independently add Gaussian noise to its local model updates, before sending the updates to the server (Geyer, Klein, and Nabi 2017). Similarly, it can also apply to our privacy protection method, by adding noise to the gradient after sparsification.

Protecting data labels. Gradient inversion can be used to

	30% SP	75% SP	95% SP	Noise
MSE ↓	0.014	0.65	2.75	1.12
PSNR ↑	155	77.9	41.8	47.8
LPIPS ↓	0.04	0.13	0.56	0.50
FID ↓	57	102	391	489
ACC↑	87.8	34.7	11.2	10.4

Table 6: The quality of recovered images with different sparsificaition rates (SR) on CIFAR-10 dataset

recover labels of client's local data (Zhu and Han. 2019b; Zhao and Bilen. 2020). As shown in Table 7, while such accuracy of label recovery can be as high as 85% if no protection method is used, applying 95% sparsification can effectively reduce such accuracy to 66.7%. Additionally, such accuracy can be further reduced to 46.4% by adding noise ($var = 10^{-3}$) to the gradient, with slight reduction (3%) of the trained model's accuracy.

Protection method	None	95% SP	95% SP+noise
Recovery accuracy	85.5%	66.7%	46.4%

Table 7: Accuracy of label recovery under different protection methods and sparsification rates (SR)

Gradient inversion should only be applied to stale clients when data and device heterogeneities are intertwined, i.e., the clients' local data is unique and unavailable elsewhere. However, to properly decide such uniqueness, the server will need to know the class labels of client's data, hence impairing the clients' data privacy. Instead, we decide the data uniqueness by comparing the directions of stale clients' model updates with the directions of other model updates from unstale clients, and only consider the stale clients' data as unique if such difference is larger than a given threshold.

We quantify such difference between model updates w_i^t, w_j^t from client i and j using cosine distance, such that

$$D_c(w_i^t, w_j^t) = 1 - w_i^t \cdot w_j^t \Big/ \|w_i^t\| \|w_j^t\|, \tag{7}$$

and the threshold is computed as the average of cosine distances between unstale model updates at $t - \tau$:

$$\frac{1}{\|S_{unstale}^{t-\tau}\|^2} \sum\nolimits_{j,k \in S_{unstale}^{t-\tau}} [D_c(w_j^{t-\tau}, w_k^{t-\tau})] \tag{8}$$

, where $S_{unstale}^{t-\tau}$ is the set of unstale clients. Since the scale of cosine distance changes during FL training (Li et al. 2023b), the average value of cosine distance adds adaptivity to the threshold.

We conducted preliminary experiments to evaluate if the server can accurately detect important model updates from unique client data. In the experiment, we emulate data heterogeneity by assigning each client with data samples from one random class, and results in Figure 8 show that the accuracy quickly grows to >90% as training progresses, and the average detection accuracy is 93%.

Epoch	20	100	200	800
Detection accuracy	74.6%	89.2%	93.7%	94.5%

Table 8: Detection accuracy from stale clients

Experiments

We evaluated our proposed technique in two FL scenarios. In the first scenario, all clients' local datasets are fixed. In the second scenario, we consider a more practical FL setting, where clients' local data is continuously updated and global data distributions are variant over time, due to dynamic changes of environmental contexts. The following baselines that tackle stale model updates in FL are used:

- **Unweighted aggregation (Unweighted)**: Directly aggregating stale model updates without applying weights.
- **Weighted aggregation (Weighted) (Shi et al. 2020)**: Applying weights to stale updates in aggregation, and weights are inversely proportional to staleness.
- **First-Order compensation (1st-Order) (Zheng et al. 2017; Zhu et al. 2022)**: Compensating errors in stale model updates using first-order Taylor expansion and Hessian approximation.
- **Future global weights prediction (W-Pred) (Hakimi et al. 2019)**: Assuming staleness as pre-known, the future global model is predicted by the first-order method above and used to compensate stale model updates.
- **FL with asynchronous tiers (Asyn-Tiers) (Chai et al. 2021)**: It clusters clients into asynchronous tiers based on staleness and uses synchronous FL in each tier.

FedAvg (Zhou and Lv. 2021) is used in all experiments for aggregating model updates. Hence, Unweighted aggregation is FedAvg with staleness, and Weighted aggregation applies extra weights to model updates in FedAvg[3]. 1st-Order, W-pred, and our method further modify such weights via compensation, and Asyn-Tiers separately uses FedAvg in each synchronous tier. The usage of FedAvg is independent from our method and other baselines, and can be replaced by other FL frameworks such as FedProx (Li et al. 2020).

For Weighted aggregation, we set the weights following (Shi et al. 2020) as $1/(1 + e^{a(\tau - b)})$, where τ is the amount of staleness and we set hyper-parameters a=0.25 and b=10 based on our experiment settings on staleness. For Asyn-Tiers, we set two asynchronous tiers and when aggregating updates of different tiers, the updates are also weighted by the number of clients in different tiers (Chai et al. 2021).

We also evaluated the performance of our technique without staleness, referred as "*Unstale*", to assess the disparity between estimated and true values of unstale model updates, as well as the impact of estimation error on FL performance.

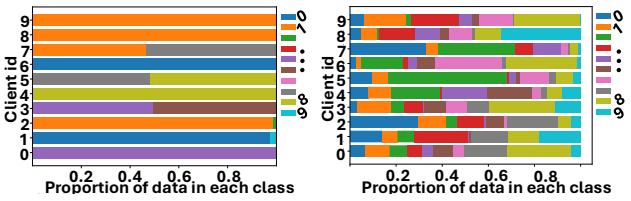
Experiment Setup

In all experiments, we consider a FL scenario with 100 clients. Each local model update on a client is trained by

[3]In FedAvg, updates are also weighted by the number of samples in clients' data, and these two weights are multiplied.

5 epochs using the SGD optimizer, with a learning rate of 0.01 and momentum of 0.5.

Data heterogeneity: We use a Dirichlet distribution to sample client datasets with different label distributions (Hsu and Brown. 2019), and use a tunable parameter (α) to adjust the amount of data heterogeneity: as shown in Figure 7, the smaller α is, the more biased these label distributions will be and the amount of data heterogeneity is higher. When α is very small, each client only has data samples of few classes.



(a) $\alpha = 0.01$ (b) $\alpha = 1$

Figure 7: Emulating data heterogeneity using the Dirichlet Distribution. Data distributions on 10 clients are shown.

Device heterogeneity: To intertwine device heterogeneity with data heterogeneity, we select one data class to be affected by staleness, and apply different amounts of staleness, measured by the number of epochs that clients' model updates are delayed, to the top 10 clients whose local datasets contain the most data samples of the selected data class. The impact of staleness can be further enlarged by applying staleness in the similar way to more data classes.

We evaluate the FL performance by assessing the trained model's accuracy **in the selected data class** being affected by staleness, and evaluate the FL training time in number of epochs. We expect that our approach can either improve the model accuracy, or achieve the similar accuracy with the baselines but use fewer training epochs.

FL Performance in the Fixed Data Scenario

In the fixed data scenario, 3 standard datasets and 1 domain-specific dataset are used in evaluations:

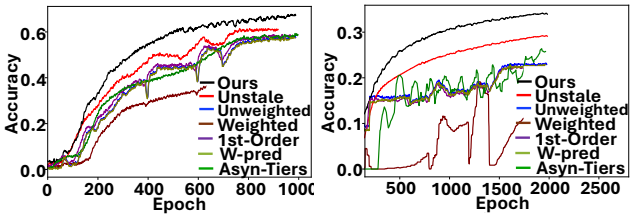
- Using MNIST (LeCun and Burges. 2010) and FMNIST (Xiao, Rasul, and Vollgraf 2017) datasets to train a LeNet model, and data class 5 is affected by staleness;
- Using CIFAR-10 (Krizhevsky 2009) dataset to train a ResNet-18 model, data class 2 is affected by staleness;
- Using a disaster image dataset MDI (Mouzannar, Rizk, and Awad 2018) to fine-tune the ResNet-18 model pre-trained with ImageNet.

The trained model's accuracies[4] using different FL schemes, with the amount of staleness as 40 epochs, are listed in Table 9. The training progresses of 1st-Order and W-Pred closely resemble that of Unweighted aggregation,

Accuracy(%)	MNIST	FMNIST	CIFAR10	MDI
Unweighted	57.4	49.2	22.8	72.3
Weighted	39.2	30.1	12.6	61.2
1st-Order	57.4	49.3	22.6	72.3
W-Pred	57.3	48.9	22.9	72.2
Asyn-Tiers	57.6	50.3	25.9	69.8
Ours	**61.2**	**55.4**	**29.4**	**75.4**

Table 9: Accuracy of the trained model with different datasets in the fixed data scenario

suggesting that estimating stale model updates with the Taylor expansion is ineffective under unlimited staleness. Similarly, Weighted aggregation will lead to a biased model with much lower accuracy. In contrast, our gradient inversion based compensation can improve the trained model's accuracy by at least 4%, compared to the best baseline. Such advantage in model accuracy can be as large as 25% when compared with Weighted aggregation. Besides image data, our method is also applicable to other data modalities such as text and time-series data. Results and discussions on these modalities with large real-world datasets are in Appendix A.



(a) MNIST, LeNet (b) CIFAR-10, ResNet18

Figure 8: The FL training procedure

Figure 8 further show the FL training procedure over different epochs, and demonstrated that our method can also improve the progress and stability of training. Furthermore, we conducted experiments with different amounts of data and device heterogeneity. Results in Tables 10 and 11 show that[5], compared with the baselines, our method can generally achieve higher model accuracy or reach the same accuracy with fewer training epochs, especially when the amount of staleness is large or the amount of data heterogeneity is high. We also use other large-scale real-world dataset to conduct experiments and results are in Appendix C.

FL Performance in the Variant Data Scenario

To continuously vary the global data distributions, we use two public datasets, namely MNIST and SVHN (Netzer 2011), which are for the same learning task but with different feature representations. Each client's local dataset is initialized as the MNIST dataset in the same way as in the fixed data scenario. Afterwards, during training, each client con-

[4]Compared to centralized training, FL models often exhibit lower accuracy, particularly under high data and device heterogeneity, as also reported in existing studies (Morafah et al. 2024).

[5]Training times in Tables 10-13 represent the relative training time required to reach convergence of the global model, assuming our method's training time is 100%.

α	1		0.1		0.01	
	Acc	Time	Acc	Time	Acc	Time
Unweighted	82.3	100	57.4	128	51.1	132
Weighted	82.4	102	39.2	171	31.1	179
1st-Order	82.5	100	57.3	129	51.5	131
W-Pred	82.8	100	57.6	126	50.9	131
Asyn-tiers	82.3	**97**	57.6	126	52.7	135
Ours	**82.3**	100	**61.2**	**100**	**58.3**	**100**

Table 10: Model accuracy and percentage of training epochs being saved, with *different amounts of data heterogeneity* controlled by α in the Dirichlet distribution. The MNIST dataset and LeNet model are used.

Staleness	10		40		100	
	Acc	Time	Acc	Time	Acc	Time
Unweighted	72.6	104	57.4	128	41.5	142
Weighted	69.4	115	39.2	171	30.5	179
1st-Order	72.6	104	57.3	129	41.8	141
W-Pred	72.6	104	57.6	126	41.7	142
Asyn-tiers	72.7	103	57.6	126	38.3	138
Ours	**73.3**	**100**	**61.2**	**100**	**47.2**	**100**

Table 11: Model accuracy and percentage of training epochs being saved, with *different amounts of staleness* measured in the number of delayed epochs.

tinuously replaces random data samples in its local dataset with new data samples in the SVHN dataset.

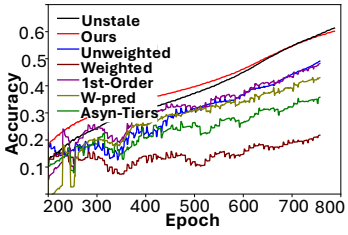


Figure 9: Model accuracy with variant data distributions in clients' local datasets

Experiment results in Figure 9 show that in such variant data scenario, since clients' local data distributions continuously change, the FL training will never converge. Hence, the model accuracy achieved by the existing FL schemes exhibited significant fluctuations over time and stayed low (<40%). In comparison, our technique can better depict the variant data patterns and hence achieve much higher model accuracy, which is comparable to FL without staleness and 20% higher than those in existing FL schemes.

We also conducted experiments with different amounts of staleness and different rates of data distributions' variations. Results in Tables 12 and 13 showed that our method outperformed the baselines with different amounts of staleness.

Related Work

Most existing solutions to staleness in FL are based on weighted aggregation (Chen and Jin. 2019; Wang 2022; Chen 2020). These existing solutions are biased towards fast clients, and will affect the trained model's accuracy when data and device heterogeneities in FL are intertwined. Other researchers suggest to use semi-asynchronous FL, where the server aggregates client model updates at a lower frequency (Nguyen 2022) or clusters clients into different asynchronous "tiers" according to their update rates (Chai 2021). However, doing so cannot completely eliminate the impact of intertwined data and device heterogeneities, because the server's aggregation still involves stale model updates.

Instead, we can transfer knowledge from stale model updates to the global model, by training a generative model and compelling its generated data to exhibit high predictive values on the original model updates (Ye 2020; Lopes, Fenu, and Starner 2017; Zhu, Hong, and Zhou 2021). Another approach is to optimize randomly initialized input data until it has good performance on the original model (YYin 2020). However, the quality and accuracy of knowledge transfer in these methods remains low, and we provided more detailed experiment results in Appendix C to demonstrate such low quality. Other efforts enhance the quality of knowledge transfer by incorporating natural image priors (Luo 2020) or using another public dataset to introduce general knowledge (Yang 2019), but require extra datasets. Moreover, all these methods require that the clients' local models to be fully trained, which is usually infeasible in FL.

Staleness	10		40		100	
	Acc	Time	Acc	Time	Acc	Time
Unweighted	60.6	99	53.2	117	39.1	131
Weighted	59.8	109	38.9	153	21.8	166
1st-Order	60.6	100	53.6	117	40.0	133
W-Pred	60.4	100	53.3	117	39.1	131
Asyn-tiers	58.2	103	46.9	118	35.7	137
Ours	**63.3**	**100**	**62.5**	**100**	**61.0**	**100**

Table 12: Model accuracy and the number of training epochs reduced, with *different amounts of staleness* measured in the number of delayed epochs

Rate	0.5		1		2	
	Acc	Time	Acc	Time	Acc	Time
Unweighted	73.1	100	39.1	131	44.1	127
Weighted	58.2	102	21.8	166	25.2	163
1st-Order	73.2	100	40.0	133	43.9	127
W-Pred	73.1	101	39.0	131	39.5	127
Asyn-tiers	68.3	98	35.7	137	39.1	130
Ours	**70.3**	**100**	**60.1**	**100**	**63.3**	**100**

Table 13: Model accuracy and the number of training epochs reduced, with *different rates of data distributions' variations* measured by number of samples changed per epoch

Conclusion

In this paper, we present a new FL framework to tackle intertwined data and device heterogeneities in FL, by using gradient inversion to estimate clients' unstale model updates. Experiments show that our technique largely improves model accuracy and reduces the amount of training epochs needed.

Acknowledgments

We thank the anonymous 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

Ahmed, L.; Ahmad, K.; Said, N.; Qolomany, B.; Qadir, J.; and Al-Fuqaha, A. 2020. Active learning based federated learning for waste and natural disaster image classification. *IEEE Access*, 8: 208518–208531.

Chai, e. a., Zheng. 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*.

Chai, Z.; Chen, Y.; Anwar, A.; Zhao, L.; Cheng, Y.; and Rangwala, H. 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.

Charles, Z.; Garrett, Z.; Huo, Z.; Shmulyian, S.; and Smith, V. 2021. On large-cohort training for federated learning. *Advances in neural information processing systems*, 34: 20461–20475.

Chen, e. a., Yujing. 2020. Asynchronous online federated learning for edge devices with non-iid data. In *2020 IEEE International Conference on Big Data (Big Data)*.

Chen, X. S., Yang; and Jin., Y. 2019. Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation. In *IEEE transactions on neural networks and learning systems*.

Chen, Y.; Yang, X.; Chen, B.; Miao, C.; and Yu, H. 2017. PdAssist: Objective and quantified symptom assessment of Parkinson's disease via smartphone. In *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 939–945. IEEE.

Geiping, e. a., Jonas. 2020. Inverting gradients-how easy is it to break privacy in federated learning? In *Advances in Neural Information Processing Systems 33*.

Geyer, R. C.; Klein, T.; and Nabi, M. 2017. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*.

Hakimi, I.; Barkai, S.; Gabel, M.; and Schuster, A. 2019. Taming momentum in a distributed asynchronous environment. *arXiv preprint arXiv:1907.11612*.

Heusel, M.; Ramsauer, H.; Unterthiner, T.; Nessler, B.; and Hochreiter, S. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30.

Hsu, H. Q., Tzu-Ming Harry; and Brown., M. 2019. Measuring the effects of non-identical data distribution for federated visual classification. In *arXiv preprint*.

Konečný, e. a., Jakub. 2016. Federated optimization: Distributed machine learning for on-device intelligence. In *arXiv preprint arXiv:1610.02527*.

Krizhevsky, a. G. H., Alex. 2009. *Learning multiple layers of features from tiny images.*

LeCun, C. C., Yann; and Burges., C. 2010. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist.

Li, T.; Sahu, A. K.; Zaheer, M.; Sanjabi, M.; Talwalkar, A.; and Smith, V. 2020. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2: 429–450.

Li, X.; Qu, Z.; Tang, B.; and Lu, Z. 2023a. Fedlga: Toward system-heterogeneity of federated learning via local gradient approximation. *IEEE Transactions on Cybernetics*, 54(1): 401–414.

Li, Z.; Lin, T.; Shang, X.; and Wu, C. 2023b. Revisiting weighted aggregation in federated learning with neural networks. *arXiv preprint arXiv:2302.10911*.

Lin, Y.; Han, S.; Mao, H.; Wang, Y.; and Dally, W. J. 2017. Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv preprint arXiv:1712.01887*.

Lopes, R. G.; Fenu, S.; and Starner, T. 2017. Data-free knowledge distillation for deep neural networks. *arXiv preprint arXiv:1710.07535*.

Luo, e. a., Liangchen. 2020. Large-scale generative data-free distillation. In *arXiv preprint arXiv:2012.05578*.

McMahan, e. a., Brendan. 2016. Communication-efficient learning of deep networks from decentralized data. In *arXiv preprint*.

Morafah, M.; Kungurtsev, V.; Chang, H.; Chen, C.; and Lin, B. 2024. Towards Diverse Device Heterogeneous Federated Learning via Task Arithmetic Knowledge Integration. *arXiv preprint arXiv:2409.18461*.

Mouzannar, H.; Rizk, Y.; and Awad, M. 2018. Damage Identification in Social Media Posts using Multimodal Deep Learning. In *ISCRAM*. Rochester, NY, USA.

Netzer, e. a., Yuval. 2011. Reading digits in natural images with unsupervised feature learning. In *Advances in Neural Information Processing Systems 32*.

Nguyen, e. a., John. 2022. Federated learning with buffered asynchronous aggregation. In *International Conference on Artificial Intelligence and Statistics. PMLR*.

Reddi, S.; Charles, Z.; Zaheer, M.; Garrett, Z.; Rush, K.; Konečnỳ, J.; Kumar, S.; and McMahan, H. B. 2020. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*.

Ro, J. H.; Breiner, T.; McConnaughey, L.; Chen, M.; Suresh, A. T.; Kumar, S.; and Mathews, R. 2022. Scaling language model size in cross-device federated learning. *arXiv preprint arXiv:2204.09715*.

Shi, G.; Li, L.; Wang, J.; Chen, W.; Ye, K.; and Xu, C. 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), 628–633. IEEE.

Tian, P.; Chen, Z.; Yu, W.; and Liao, W. 2021. Towards asynchronous federated learning based threat detection: A DC-Adam approach. *Computers & Security*, 108: 102344.

Wang, e. a., Qiyuan. 2022. AsyncFedED: Asynchronous Federated Learning with Euclidean Distance based Adaptive Weight Aggregation. In *arXiv preprint*.

Wu, Y.; Zhang, S.; Yu, W.; Liu, Y.; Gu, Q.; Zhou, D.; Chen, H.; and Cheng, W. 2023. Personalized federated learning under mixture of distributions. In *International Conference on Machine Learning*, 37860–37879. PMLR.

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

Xie, S. K., Cong; and Gupta., I. 2019. Asynchronous federated optimization. In *arXiv preprint*.

Yang, e. a., Ziqi. 2019. Neural network inversion in adversarial setting via background knowledge alignment. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security.*

Ye, e. a., Jingwen. 2020. Data-free knowledge amalgamation via group-stack dual-gan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Yin, e. a., Hongxu. 2020. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Yin, e. a., Hongxu. 2021. See through gradients: Image batch recovery via gradinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

YYin, e. a., Hongxu. 2020. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.*

Zhang, e. a., Richard. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Zhao, e. a., Yue. 2018. Federated learning with non-iid data. In *arXiv preprint*.

Zhao, K. R. M., Bo; and Bilen., H. 2020. idlg: Improved deep leakage from gradients. In *arXiv preprint arXiv:2001.02610*.

Zheng, S.; Meng, Q.; Wang, T.; Chen, W.; Yu, N.; Ma, Z.-M.; and Liu, T.-Y. 2017. Asynchronous stochastic gradient descent with delay compensation. In *International Conference on Machine Learning*, 4120–4129. PMLR.

Zhou, e. a., Chendi. 2021. TEA-fed: time-efficient asynchronous federated learning for edge computing. In *Proceedings of the 18th ACM International Conference on Computing Frontiers*.

Zhou, Q. Y., Yuhao; and Lv., J. 2021. Communication-efficient federated learning with compensated overlap-fedavg. In *IEEE Transactions on Parallel and Distributed Systems*.

Zhou, Y.; Ye, Q.; and Lv, J. 2021. Communication-efficient federated learning with compensated overlap-fedavg. *IEEE Transactions on Parallel and Distributed Systems*, 33(1): 192–205.

Zhu, H.; Kuang, J.; Yang, M.; and Qian, H. 2022. Client Selection With Staleness Compensation in Asynchronous Federated Learning. *IEEE Transactions on Vehicular Technology*, 72(3): 4124–4129.

Zhu, Z.; Hong, J.; and Zhou, J. 2021. Data-free knowledge distillation for heterogeneous federated learning. In *International conference on machine learning*, 12878–12889. PMLR.

Zhu, Z. L., Ligeng; and Han., S. 2019a. Deep leakage from gradients. In *Advances in neural information processing systems*.

Zhu, Z. L., Ligeng; and Han., S. 2019b. Deep leakage from gradients. In *Advances in neural information processing systems 32*.