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

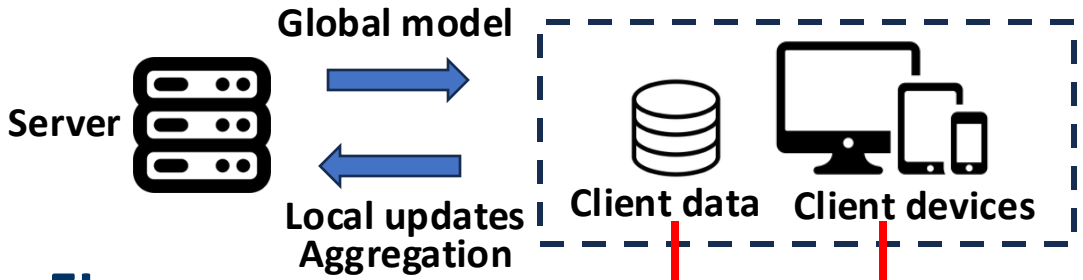
AAAI 25

Haoming Wang (hw.wang@pitt.edu)
Wei Gao (weigao@pitt.edu)

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

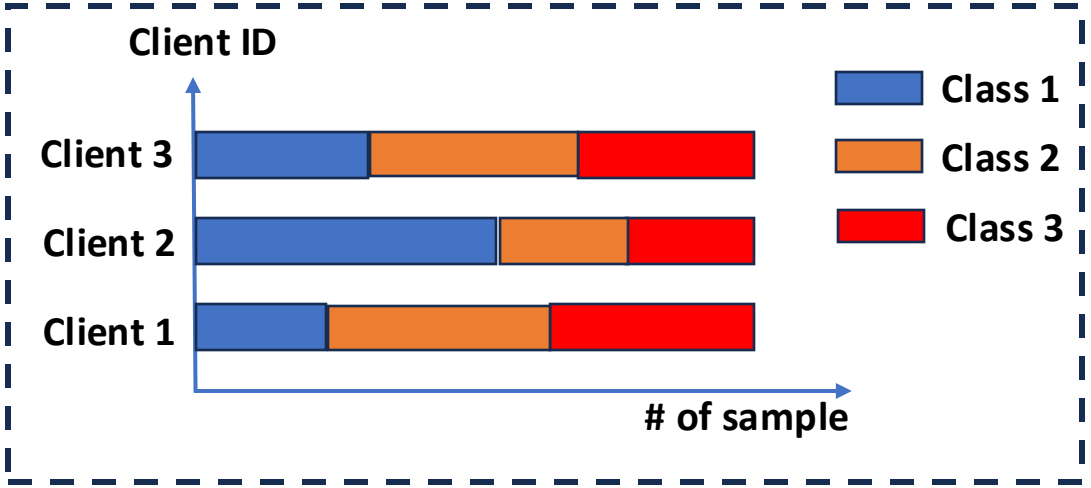
Introduction to intertwined heterogeneities in FL

Federated learning (FL) :



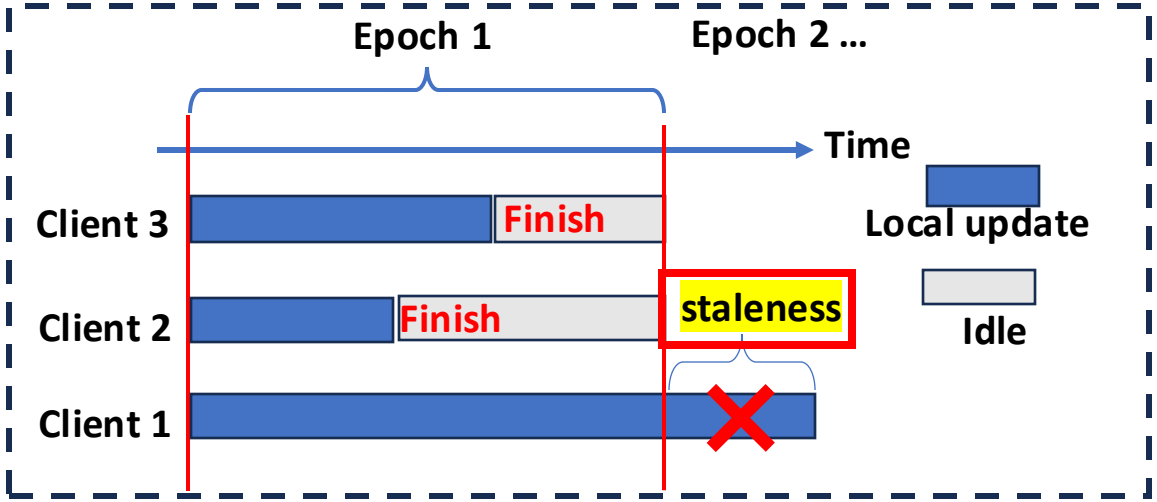
System challenge in FL:

Data heterogeneity



Reduce model accuracy

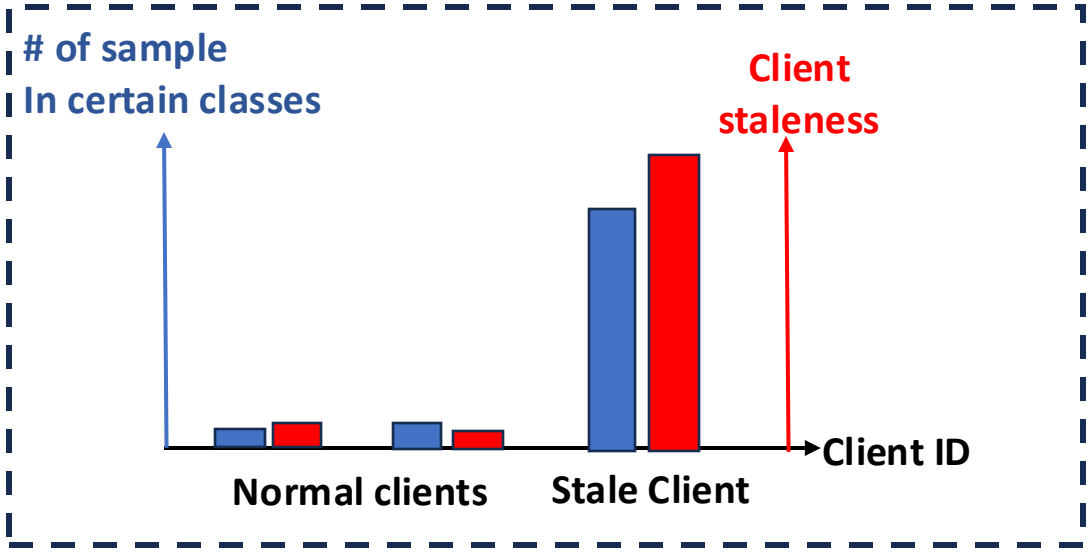
Device heterogeneity



Slow down the training

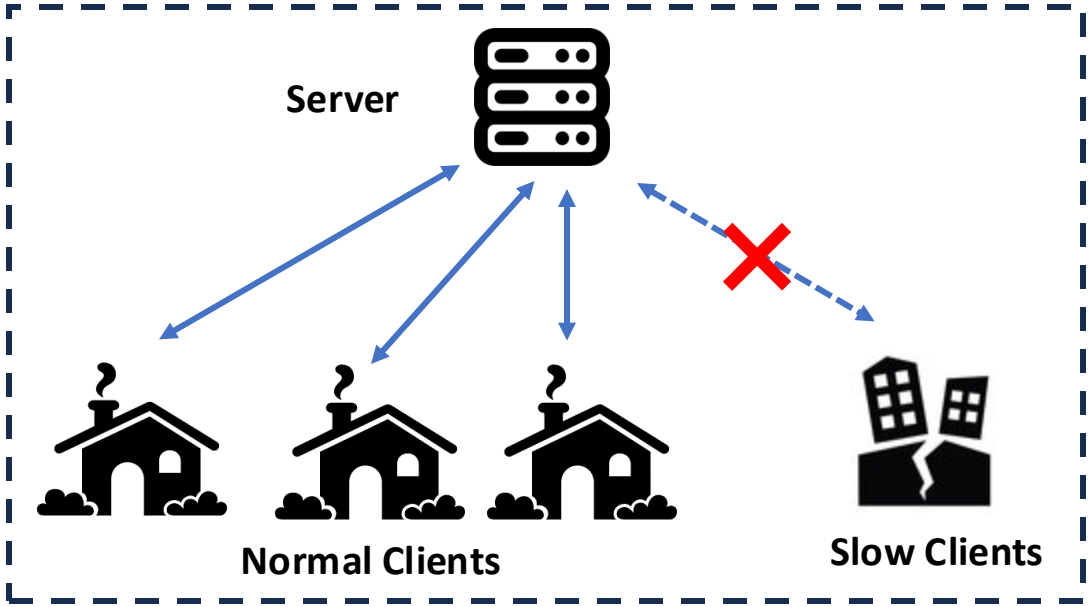
Intertwined (data and device) heterogeneities:

Correlated Data distribution and Staleness



e.g. data in certain classes or with particular features may only be available at some slow clients

A real-world scenario of intertwined heterogeneities

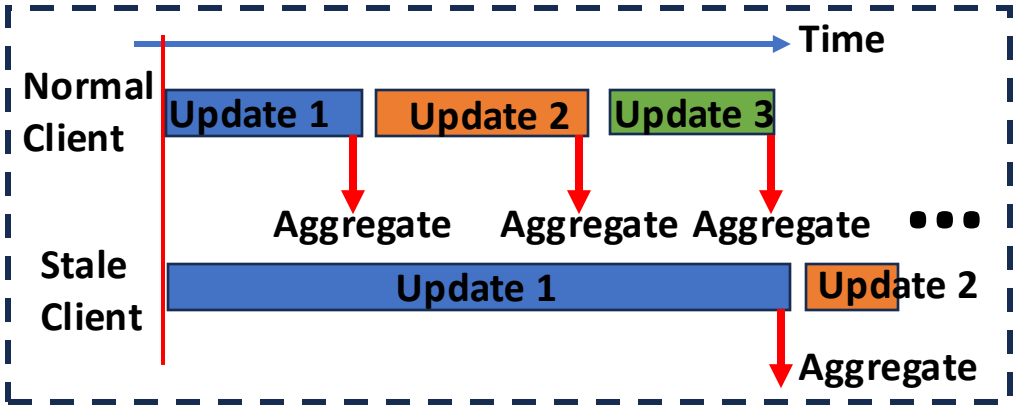


Hazard rescue

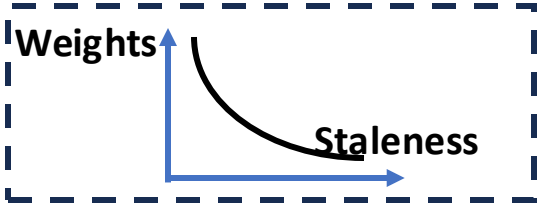
Limitation of existing methods

Methods 1: Asynchronous FL with weighted aggregation

Asynchronous FL (AFL)[1] :



Weighted aggregation[2]:

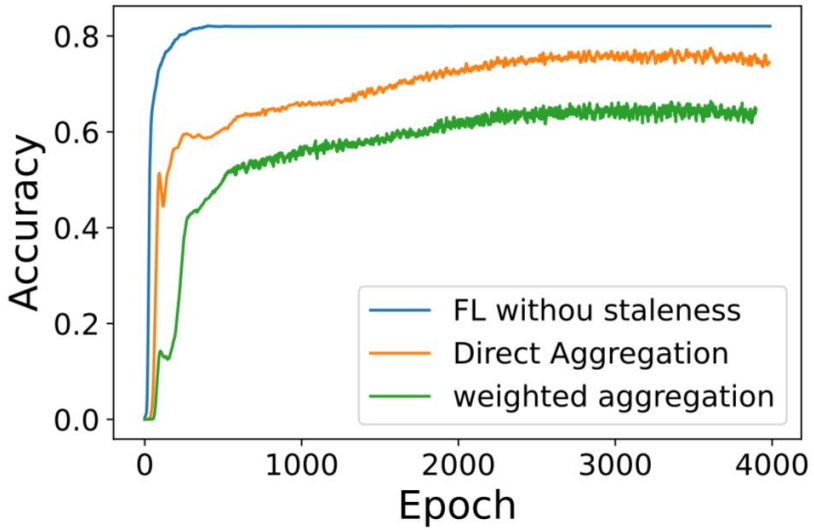


Reduced weights to stale updates

Limitation :

Data on stale clients contribute less

Biased model:



Introduction to intertwined heterogeneities

Methods 2:Convert a stale update into an unstable one

First order compensation [3]:

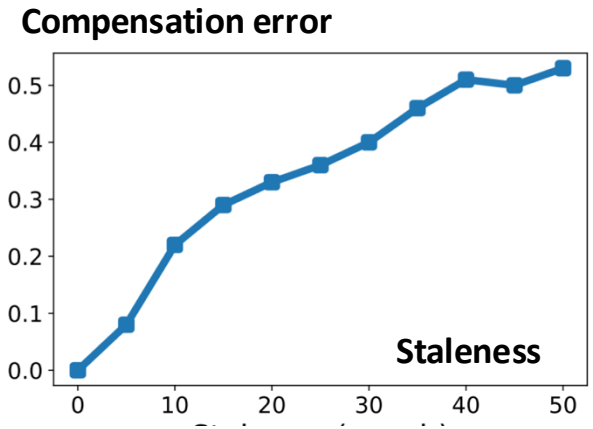
$$g(w_{t+\tau}) \approx g(w_t) + \nabla g(w_t)(w_{t+\tau} - w_t)$$, (τ is staleness)

estimated update — stale update — Correction term

use Taylor expansion to estimate the unstable update

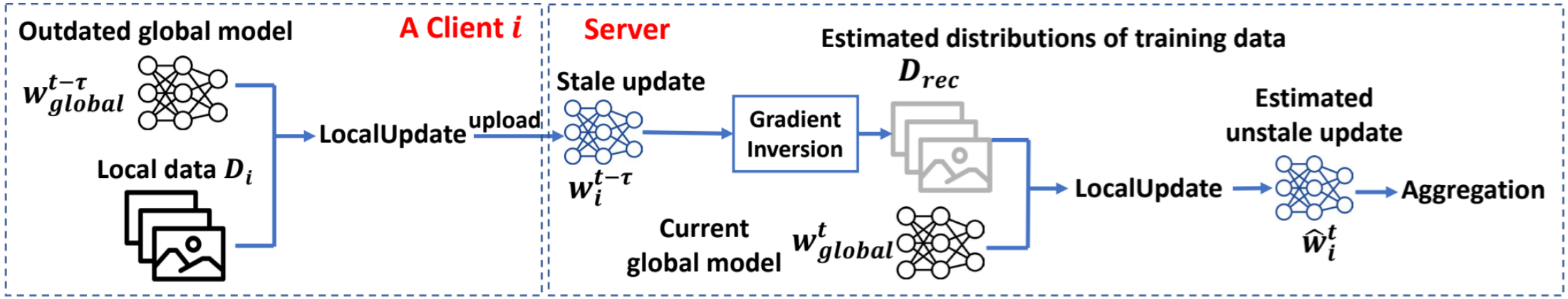
Limitation:

Compensation error will significantly increase with staleness:



Our method: gradient inversion based compensation

Main idea: convert the stale update to unstable using gradient inversion



Step 1: local data estimation

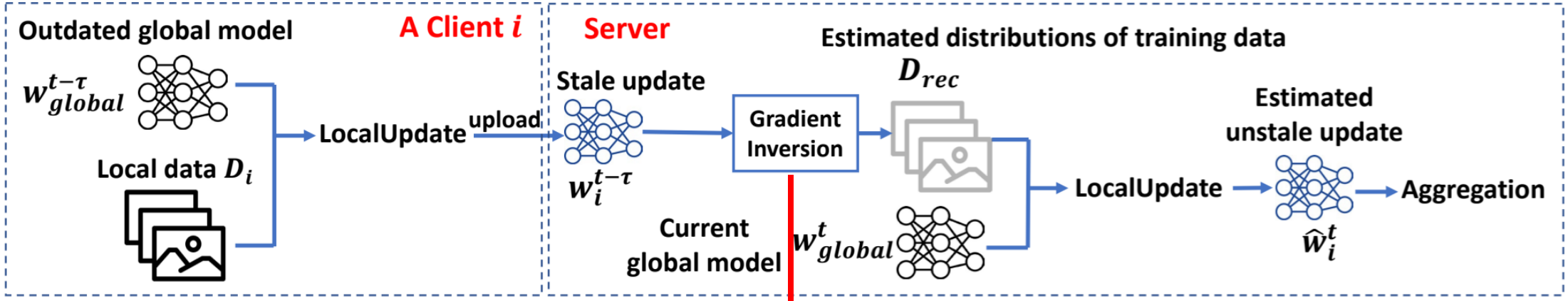
the server estimate client's data distribution(D_{rec}) with gradient inversion

Step 2: unstale update estimation

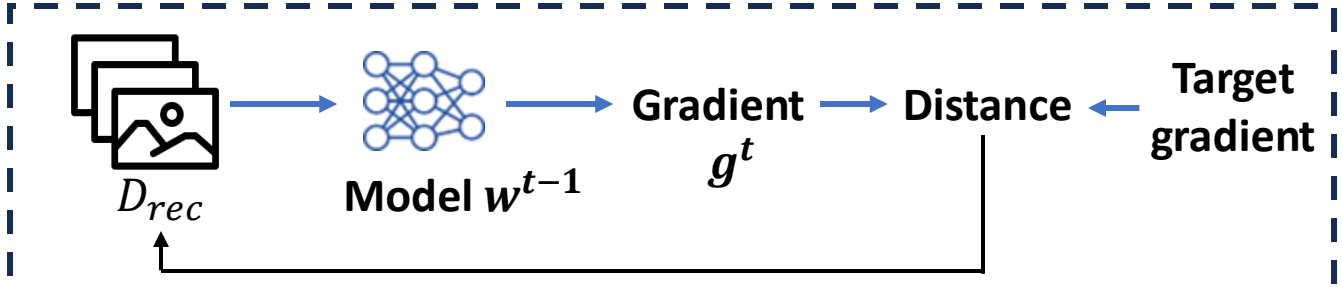
use D_{rec} to retrain the current global model as the estimation of unstale update

Our method: gradient inversion based compensation

Main idea: convert the stale update to unstable using gradient inversion



Step 1: local data estimation

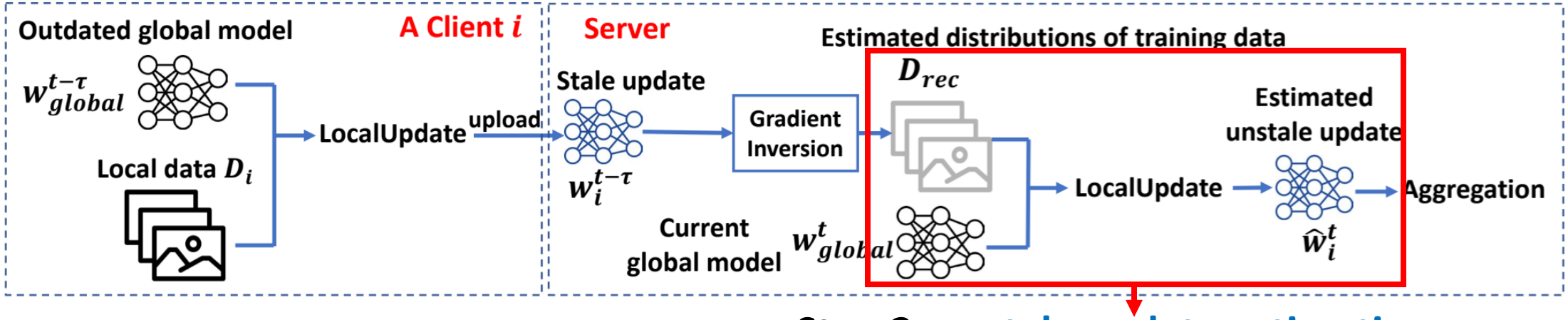


Minimize the distance by optimizing D_{rec}

$$D_{rec}^{*} = arg\,min_{D_{rec}} \left\| \frac{\partial L[D_{rec}; w^{t-1}]}{\partial w^{t-1}} - g^{t} \right\|_2^2$$

Our method: gradient inversion based compensation

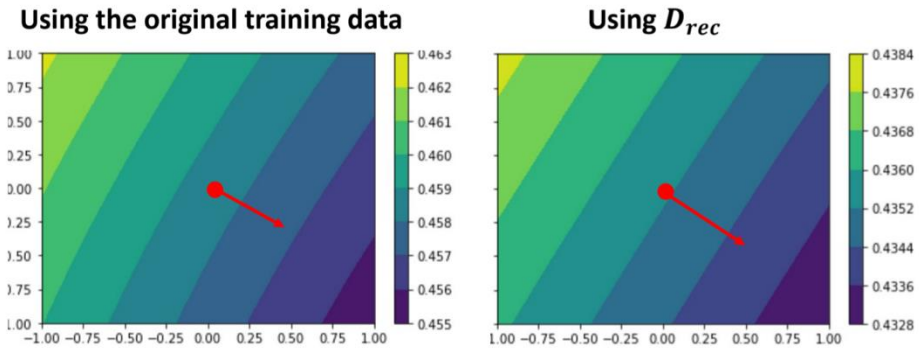
Main idea: convert the stale update to unstable using gradient inversion



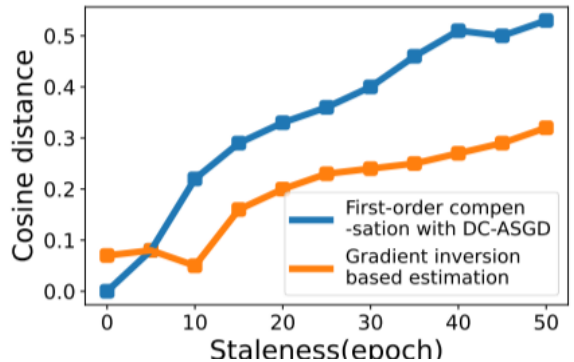
Step 2: unstale update estimation

Rationale of using D_{rec} to estimate the unstale update :

Similar loss surface compared with the original data



Unstale update estimation error:



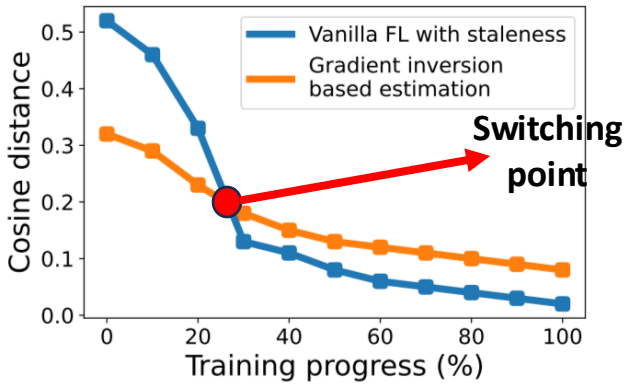
First order compensation

Gradient inversion based compensation

Our method: gradient inversion based compensation

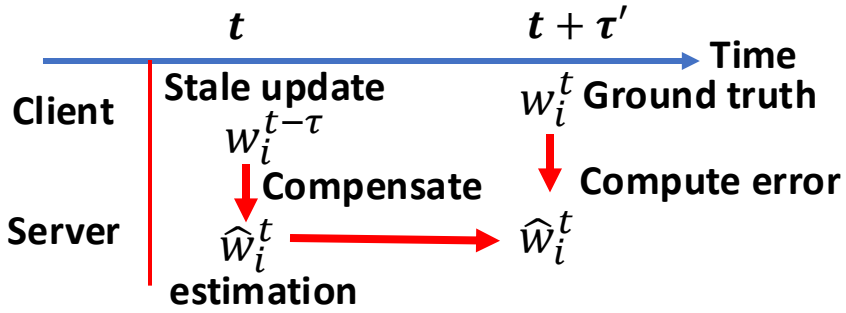
Details of method design 1: Switching back to Vanilla FL in Later Stages of FL Training

Vanilla FL has less error as model converges:



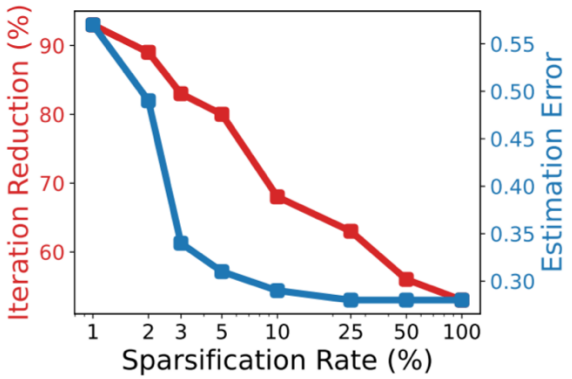
Deciding the switching point:

Computing the current error at later epoch

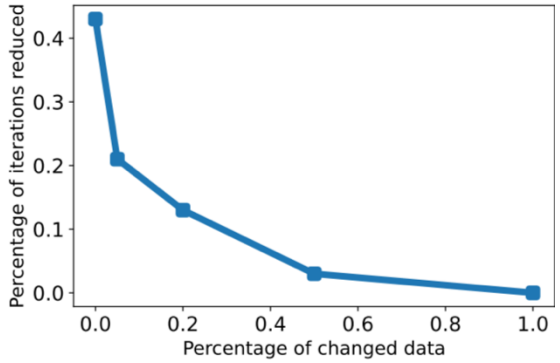


Details of method design 2: Reducing the Computing Cost of Gradient Inversion

Sparsification: reduce the the objective function complexity



Initialize D_{rec} with previous recover results



Our method: gradient inversion based compensation

Details of method design 3: protecting Client data privacy

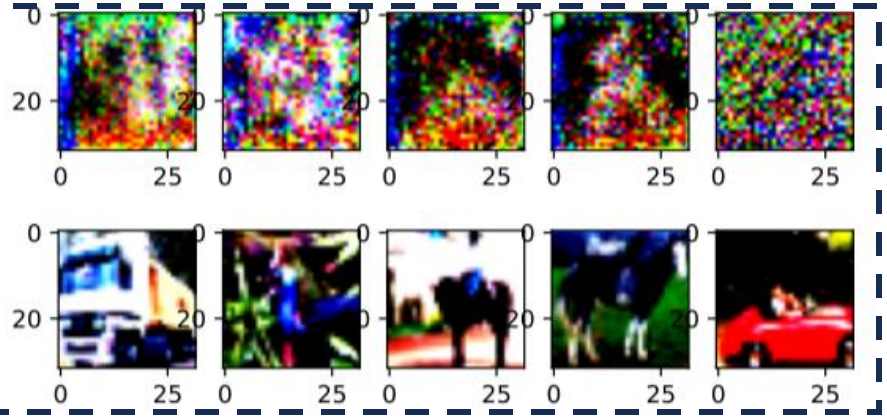
Most FL scenarios:

each client has a large batch of samples

Nearly impossible to pixel-wisely recover :

Samples in D_{rec}

Raw data



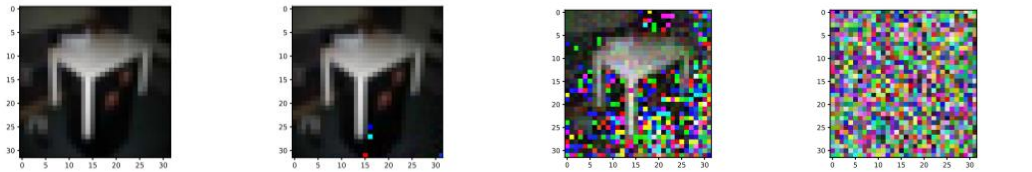
5 best matches between samples in D_{rec} and raw data (using LPIPS score as image similarity)

Extreme scenarios:

each client only has one sample)

- Use sparsification and gradient noise to mitigate the attack power of gradient inversion

Protect the input image:



(a) Original image (b) 0% sparsification (c) 30% sparsification (d) 95% sparsification

Protect the label:

Defense	None	95% sparsification	95% sparsification + noise
Label recovery ACC	85.5%	66.7%	46.4%

Experiment results

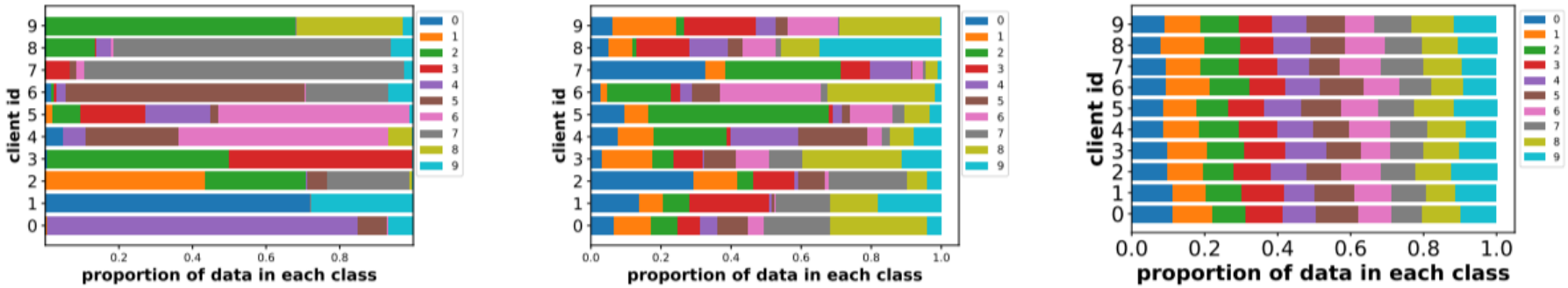
Baselines:

- Unweighted aggregation (Vanilla FL)
- Weighted aggregation[2]
- First order compensation (1st-order)[3]
- Future model prediction (W-pred)[5]
- FL with asynchronous tiers (Asyn-tiers)[6]

Experiment setting:

Data heterogeneity: sample different label distribution using Dirichlet distribution

Dirichlet distribution with different α:



(a) $\alpha = 0.1$ (b) $\alpha = 1$ (c) $\alpha = 100$

Device heterogeneity (intertwined with data heterogeneity):

select one data class to be affected by staleness, and apply different amounts of staleness to 10 clients With the most data samples in the class

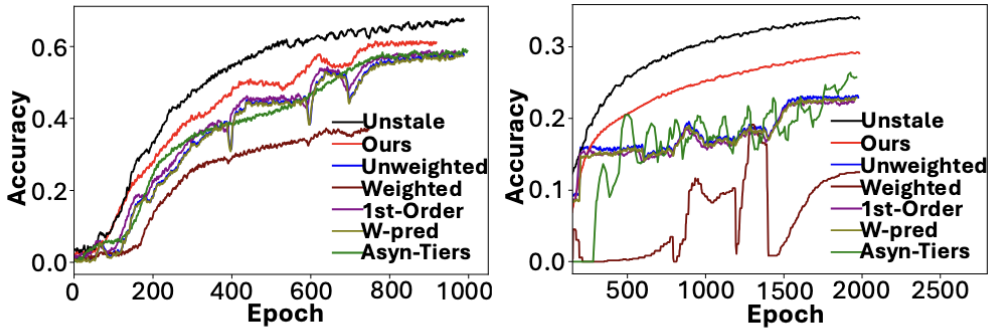
Experiment results

FL Performance in the Fixed Data Scenario

Main results:

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**

Model accuracy with different datasets



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

Accuracy curve during training

Variations:

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**

Performance under different data heterogeneity

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**

Performance under different staleness

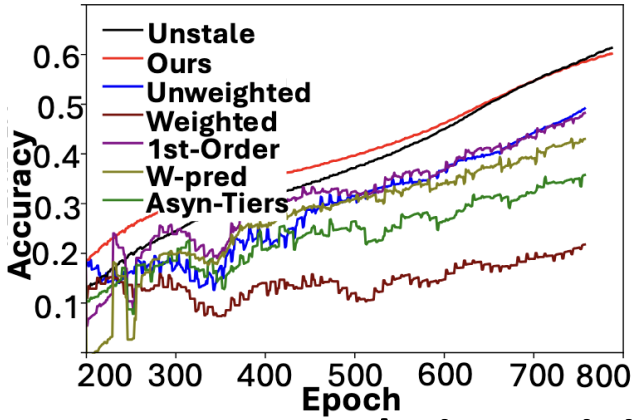
Experiment results

FL Performance in the Variant Data Scenario

Variant data setting:

- Client data is initialized with MNIST data
- During training MNIST samples are gradually replaced by SVHN samples

Main results:



Accuracy curve during training

Variations:

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**

Performance under different staleness

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**

Performance under different data variation rates

References

[1]Chen, Yujing. Asynchronous online federated learning for edge devices with non-iid data. In 2020 IEEE International Conference on Big Data (Big Data), 2020.
[2]Wang, Qiyuan. AsyncFedED: Asynchronous Federated Learning with Euclidean Distance based Adaptive Weight Aggregation. In arXiv preprint, 2022.
[3] S. Zheng, Q. Meng, T. Wang, W. Chen, N. Yu, Z.-M. Ma, and T.-Y. Liu. Asynchronous stochastic gradient descent with delay compensation. In International Conference on Machine Learning, pages 4120–4129. PMLR, 2017.
[4] Z. L. Ligeng Zhu and S. Han. Deep leakage from gradients. In Advances in neural information processing systems 32, 2019
[5] I. Hakimi, S. Barkai, M. Gabel, and A. Schuster. Taming momentum in a distributed asynchronous environment. arXiv preprint arXiv:1907.11612, 2019.
[6] Chai, Zheng. 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, 2021.