ECE 273 Project Report: Federated Learning

Yein Kim
Department of Electrical and Computer Engineering,
University of California, San Diego
y5kim@ucsd.edu

1. Objectives

These days, the widespread usage of computing devices, such as mobile phones and wearable device, has increased the amount of private data. This phenomenon poses a problem: given data scattered across multiple edge devices, how can we train one common model effectively while ensuring the data privacy?

One proposed solution is federated learning, a learning mechanism where each edge user trains its model and the server aggregates the local model update to train the global model. This new global model parameter is then propagated to the edge users. Hence, the server and edge users exchange model parameters instead of the whole data.

In this project, we aim to study the applicability and challenges of federated learning in image classification. We first experiment whether a federated learning model converges as fast as a centralized learning model under different configurations. Then we investigate two challenges: presence of malicious edge users and possible leak of private data.

2. Background and Motivation

2.1. Why federated learning?

Federated learning is important and highly applicable in two aspects [3].

1. Privacy sensitive and highly distributed data: Federated learning doesn't require the exchange of private data between edge users and the server. Thus, it can be used under strict privacy practices. For example, it has been applied to improve the next word prediction models in Google's Gboard [4] without accessing users' text messages directly. It is also considered when modeling on healthcare data [11] and can even be combined with blockchain technology to train models in a fully decentralized fashion while protecting the data privacy [9].

2. Limited communication networks: Federated learning only requires the exchange of model parameters between the server and edge users. Since model parameters tend to be more light-weighted than full data, federated learning alleviate the communication burden. Hence, it can be useful where communication networks are challenging, such as underwater [7] and unmanned aerial vehicle networks [2].

Along with these use cases, growing storage and computing power of devices enable the effective training on edge devices [8]. These resources increase the feasibility of federated learning along with the importance of studying it.

2.2. Datasets and models

In this project, we use two datasets: Fashion-MNIST [10] and CIFAR10 [6]. CIFAR10 is only used when implementing "deep leakage from gradients."

- Fashion-MNIST is a set of Zalando's grey-scale images of 10 classes with dimension of $1 \times 28 \times 28$. It consists of 60,000 training images and 10,000 test images. We train a three-layer Convolutional Neural Network (CNN) that consists of two convolutional layers with the sigmoid activation functions and one fully connected output layer on this dataset.
- CIFAR-10 is a set of RGB images of 10 classes with dimension of $3 \times 32 \times 32$. It includes 50,000 training images and 10,000 test images. We train a lightweight ResNet-18 CNN model [5] where ReLU activation functions are all replaced with sigmoid activation functions.

3. Methods

3.1. Federated learning model formulation

We assume that there are one central server and M edge users. In federated learning, all users share parameters with the global model $\theta \in \mathbb{R}^d$. Each user's local model parameter is represented as $\theta_i \in \mathbb{R}^d \; \forall i \in \{1, 2, \cdots, M\}$. We also suppose that there are N training images and each user has its own local data, $D^i = \{(x_n^i, y_n^i)\}_{n=1}^{n_i}$ ($\sum_i |D^i| = N$). We use cross entropy loss as the loss function, $L(x_n^i, y_n^i; \theta^i)$ throughout the project.

1. Local training: Each user i optimizes the objective function,

$$f(\theta^i, D^i) = \frac{1}{n_i} L(x_n^i, y_n^i; \theta^i)$$

   In other words, each user minimizes the average cross entropy over its local training data. Note that the user can either train the model iteratively with stochastic gradient descent (SGD) or compute the loss gradient directly. At this stage, we can configure local data distribution, training iterations and learning rate.

2. Global aggregation: The server updates the global model parameter with the objective function, $F(\theta)$. Here, we can vary the aggregation frequency (i.e., the number of local training iterations before each aggregation epoch) and the ratio of local models selected for aggregation.

Now we consider and implement two methods for global aggregation.

3.2. Aggregation1: Federated averaging

At each aggregation epoch t, we update the global model θ_t as the weighted average of the local updates:

$$\theta_{t+1} = \frac{M}{|g_t|} \sum_{i \in g_t} p_i \theta_{t+1}^i$$

where $p_i = \frac{n_i}{N}$, the fraction of each user's local training data and g_t is the set of local models chosen for aggregation at epoch t.

3.3. Aggregation2: Byzantine-robust aggregation

Attack scenario. Suppose that $\alpha\%$ of edge users are Byzantine, meaning that they train on "poisoned" data - each training label y is replaced with $9 - y$.

The goal is to update the global model robust to such a Byzantine attack. Yin, Chen, Kannan and Bartlett suggest two update algorithms, proven to be statically optimal given strongly convex losses [12].

Gradient descent algorithm. At each aggregation epoch t,

- Local update $\delta^i_{t+1} = \theta^i_t - \theta^i_{t+1}$
- Global update: there are two options. The server can take either the coordinate-wise median or the trimmed mean of the local updates with a parameter β.

$$\delta_{t+1} = \begin{cases} med\{\delta^i_{t+1}\} & \text{for } i \in g_t \text{ (1)} \\ trmean_\beta\{\delta^i_{t+1}\} & \text{for } i \in g_t \text{ (2)} \end{cases}$$

- New global model parameter $\theta_{t+1} = \theta_t - \eta\delta_{t+1}$ where η is the learning rate of the global training.

Trimmed mean is the mean computed after removing $\frac{\beta}{2}\%$ of the largest and $\frac{\beta}{2}\%$ of the smallest elements. In the experiment, we assume the complete knowledge, letting $\beta = \alpha$. Also, note that each user computes the loss gradient directly based on its training images.

One-round algorithm.

- Local parameter $\theta^i = argmin_\theta f(\theta^i, D^i)$
- Global parameter $\theta = med\{\theta^i\}$

In this case, each user updates the model based on SGD with 500 iterations, learning rate of 0.01 and momentum of 0.9. Each batch consists of 10% of the local training images.

Note that the main difference between gradient descent and one-round algorithms is the aggregation frequency. Global model is updated after each local training iteration in the former while it is updated only once at the end in the latter.

3.4. Deep leakage from gradients

Given a local gradient, this method aims to recover a batch of training data (x, y). The idea is to update dummy data iteratively until its gradient with respect to the global model parameter is as close to the local gradient as possible. To formulate mathematically,

- Given a global parameter $\theta \in \mathbb{R}^d$ and a local gradient $\nabla\theta = \frac{\partial L(x,y;\theta)}{\partial\theta}$
- Initialize dummy training data (x', y')
- Compute the dummy gradient $\nabla\theta = \frac{\partial L(x',y';\theta)}{\partial\theta}$
- $x'^*, y'^* = argmin_{x',y'} ||\nabla\theta' - \nabla\theta||^2$

4. Results

4.1. Aggregation1: Federated averaging

Experiment setting. We experiment on F-MNIST. There are 20 epochs in total. One epoch corresponds to a training iteration in centralized learning and a global aggregation in federated learning. We train each local model using SGD with 500 iterations, batch size of 20 images, learning rate of 0.01 and momentum of 0.9. We vary two configurations: local data distribution and total number of edge users M.

- Local data distribution: In iid setting, $|D^i| = \frac{N}{M}$, each user i has the same training data size with images of each class uniformly distributed. In non-iid setting, user data are distributed by Dirichlet distribution with the concentration hyperparameter 0.9 for each class. Hence, images of each class can be concentrated in some users.

- Total number of edge users $M = 100, 20, 10$. We consistently choose 10 users per epoch so that the fraction of local models aggregated is $10\%, 50\%$ and 100% respectively.

Test accuracy. The centralized learning models achieves the final test accuracy of 87.74%. In Table 1, we can observe that the federated learning model test accuracy is slightly lower than, but comparable to the centralized learning model's. In particular, when local data are distributed in iid fashion, the test accuracy is consistent around 86%, regardless of the fraction of models aggregated. This implies that a federated learning model can be trained very efficiently, requiring only a small fraction of local updates.

% aggregated models	iid	non-iid
10	86.32	84.21
50	86.75	83.15
100	86.17	86.29

Table 1: Final test accuracy of federated learning model with different local data distributions and fractions of aggregated models.

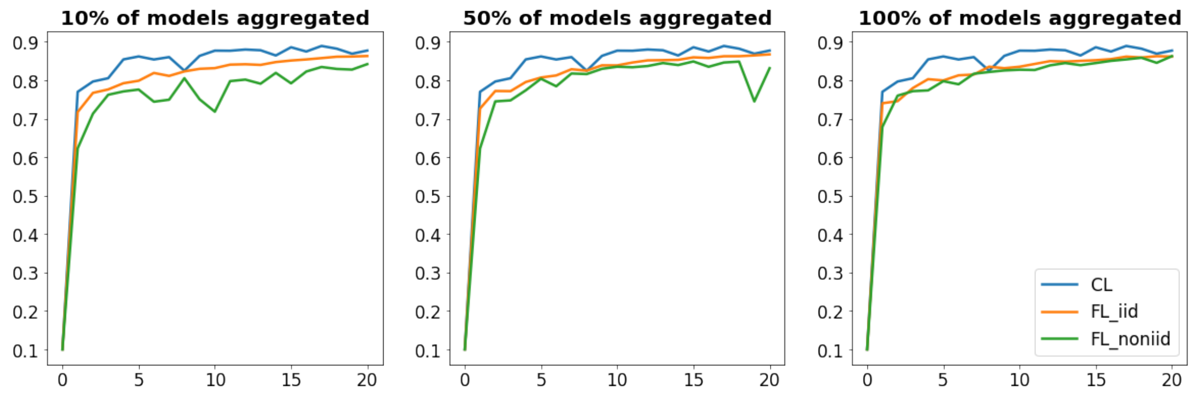


Figure 1: Test accuracy of centralized learning model, federated learning models trained on iid and non-iid local data distributions over 20 epochs. We select $10\%, 50\%, 100\%$ of users for global aggregation respectively.

On the other hand, when local data are non-iid distributed, a federated learning model tends to converge faster with a higher fraction of aggregated models as can be shown in Figure 1.

4.2. Aggregation2: Byzantine-robust aggregation

Experiment setting. We experiment on F-MNIST. In total, there are $M = 20$ users, all of whom participate in aggregation. Local data are distributed in iid fashion. There are 500 local iterations in total - 500 aggregation epochs in gradient descent algorithm and 1 aggregation epoch in one-round algorithm. We vary the Byzantine ratio as 0, 0.1, 0.2 and 0.4.

Test accuracy. We first observe the performance of the gradient descent algorithm in Figure 2. Overall, the plots are very noisy as the global model is updated stochastically. This implies that federated learning may be more effective when edge users train their models over a few iterations rather than taking the loss gradients at one shot.

We can see an interesting pattern in the federated averaging's test accuracy (blue line). It usually converges slowly in the beginning. Then there is a break point where its test accuracy jumps up. As the Byzantine ratio increases, this break point is shifted towards the right. In other words, the Byzantine attack is effective on the federated averaging model in the early training stage. On the other hand, both median and trimmed mean models' convergence behaviors are consistently smooth.

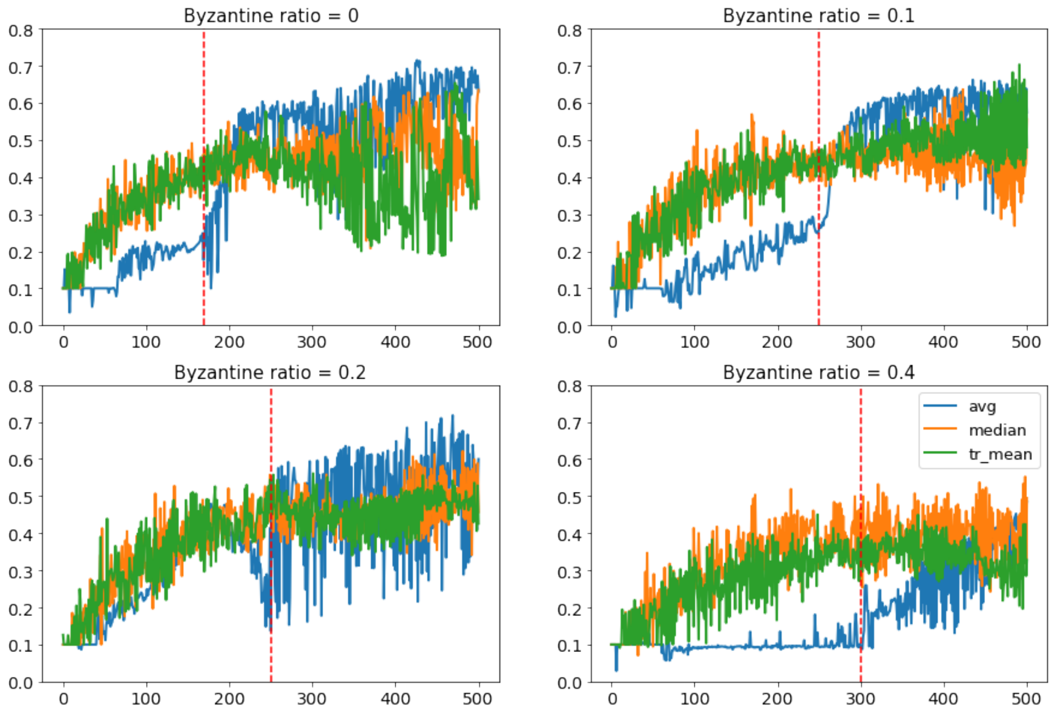


Figure 2: Test accuracy of federated learning models aggregated with weighted average, median and trimmed mean with gradient descent across different values of Byzantine ratio.

The Byzantine-robust aggregation method shines in one-round algorithm. In table 2, we can clearly see that the federated averaging's final test accuracy decreases as the Byzantine ratio increases. The median aggregated model's accuracy, on the other hand, is consistent around 70%, with a slight decline even with 40% of Byzantine users. Therefore, we can conclude that Byzantine-robust aggregation is effective with sufficiently trained local models' updates while the performance of federated averaging degrades with a high fraction of Byzantine users.

Byzantine ratio	FedAvg	CooMedian
0	72.91	72.62
0.1	71.41	72.66
0.2	61.01	72.37
0.4	**40.81**	**70.37**

Table 2: Final test accuracy of federated learning models aggregated with federated averaging and coordinate-wise median in one-round over different fractions of Byzantine users.

4.3. Deep leakage from gradients

Experiment setting. We experiment on F-MNIST and CIFAR-10. Each user computes and sends the loss gradient and the server aggregates the local updates after each iteration. We follow the original paper [13] closely - dummy images are updated with L-BFGS and learning rate 1 iteratively. We apply this method on models whose weights are randomly initialized and configure the batch size to be 1 and 2.

Simulations. We simulate the algorithm on both F-MNIST and CIFAR-10 where batch size $= 1, 2$ with 100 iterations. When there is only one image per batch, we can recover the image easily on both datasets. In figure 3, we can observe that the F-MNIST dummy image already converges with 20 iterations although it does not match the original image exactly. It takes more iterations up to 60 for the CIFAR-10 dummy image to converge. However the dummy image resembles the original more closely on CIFAR-10 as can be seen in figure 4.

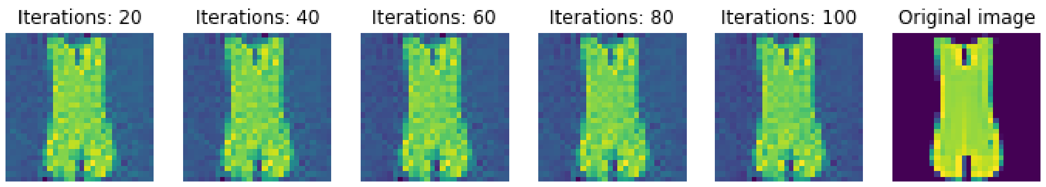


Figure 3: F-MNIST image recovered with 100 iterations

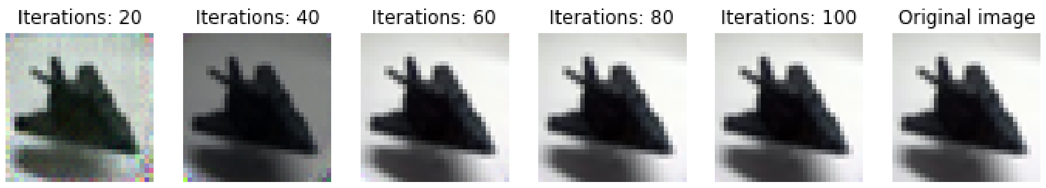
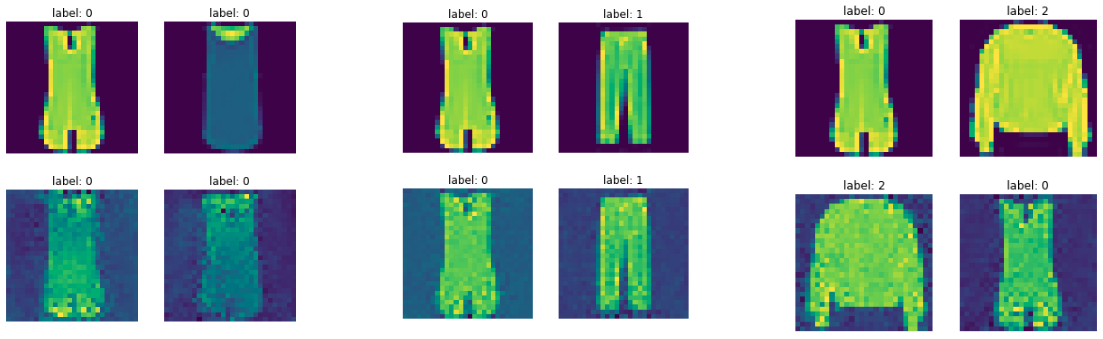


Figure 4: CIFAR-10 image recovered with 100 iterations

We have conflicting results when there are two images per batch. On F-MNIST, images from the same class are harder to recover than those from different classes as shown in figure 5. On the other hand, images from the same class were all recovered on CIFAR-10 over ten experiments. However, there are various outcomes when images are from two different classes. In figure 6b, we can observe that the recovered plane image resembles the original even more closely than in figure 6a. In figure 6c, images are not recovered while in figure 6d, neither images nor labels are recovered. More investigations are necessary to understand the divergent behaviors of the algorithm on different datasets and batches. Possible contributing factors can be the model architecture/convergence or the diversity of images in a batch.

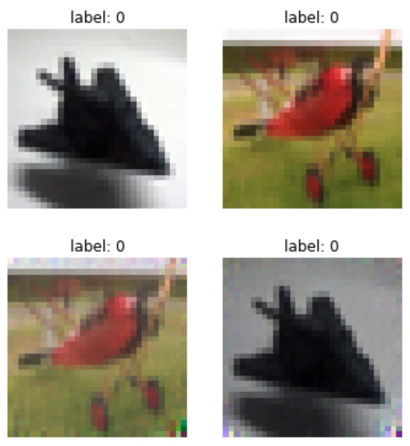


(a) same class

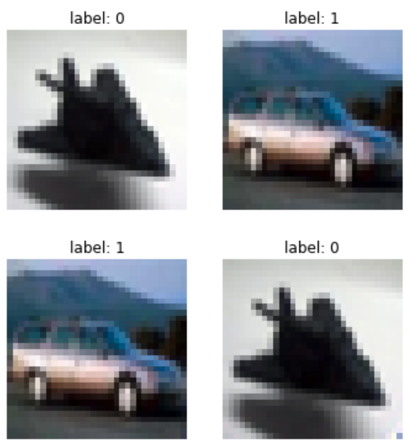
(b) different classes

(c) different classes

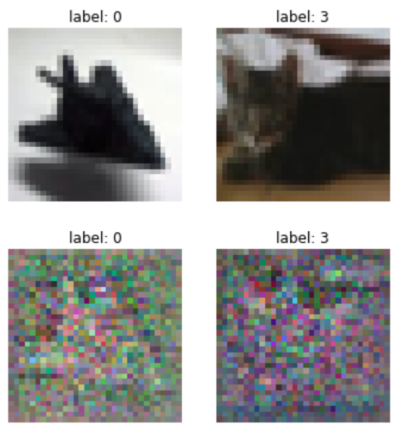
Figure 5: Two F-MNIST images per batch recovered with 100 iterations



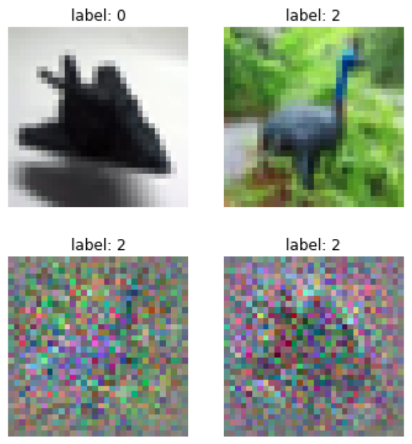
(a) Successful recovery of images and labels from the same class



(b) Successful recovery of images and labels from different classes



(c) Unsuccessful recovery of images from different classes



(d) Unsuccessful recovery of images and labels from different classes

Figure 6: Two CIFAR-10 images per batch recovered with 100 iterations

5. Discussion

Overall, we learned that a federated learning model converges as fast as a centralized learning model on F-MNIST. The federated learning model's convergence tends to slow down when local data are non-iid distributed and a small fraction of local models are selected for aggregation. However, when local data are iid distributed, it can achieve a consistently high test accuracy even with 10% of users selected for aggregation. This may enable the server to update the global model efficiently based on the updates from a few users.

Despite this promising result, federated learning faces two main challenges. First, some local updates may be polluted. For example, a fraction of users train their models on intentionally modified training data and their local updates can confuse the global model. We implement two Byzantine-robust update algorithms - gradient descent and one-round - based on coordinate-wise median and trimmed mean. The one-round algorithm is effective on F-MNIST, achieving a consistent test accuracy over different Byzantine ratios. Both median and trimmed mean aggregations manifest more stable and consistent convergence behaviors than the federated averaging method. However, they are still noisy, taking a lot of iterations to achieve a descent test accuracy. This implies that federated learning, in general, may work better when local models are trained stochastically over a sufficient number of iterations rather than when the loss gradient is directly computed. Moreover, Byzantine-robust algorithms assume convex loss functions. Their performances may degrade when tested with non-convex loss functions. For future work, we can extend our experiment to a more complex dataset such as CIFAR-10 and a more subtle attack such as model poisoning attack [1] to validate the robustness of these algorithms.

Finally, we investigate the other challenge: potential leak of training data. Deep leakage from gradients algorithm successfully recovers images from both F-MNIST and CIFAR-10 when there is only one image per batch. However, we have conflicting results when there are two images per batch. Images from different classes are recovered more easily on F-MNIST while images from the same class recovered better on CIFAR-10. In particular, the quality of recovered images and labels vary when they belong to different classes. This result raises a question on the robustness of the algorithm. Its performance may depend on the model architecture or the diversity of images in a training batch. Moreover, the algorithm has a limited applicability as it requires the model to be twice-differentiable when updating the dummy data with gradient-based methods. When testing the algorithm on the original ResNet-18 with ReLU activation functions, we failed to recover CIFAR10 images. Therefore, more research is needed to improve the robustness and applicability of the deep leakage from gradients.

References

[1] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo. Analyzing federated learning through an adversarial lens. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 634–643, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[2] B. Brik, A. Ksentini, and M. Bouaziz. Federated learning for uavs-enabled wireless networks: Use cases, challenges, and open problems. *IEEE Access*, 8:53841–53849, 2020.

[3] T. Gafni, N. Shlezinger, K. Cohen, Y. C. Eldar, and H. V. Poor. Federated learning: A signal processing perspective. *CoRR*, abs/2103.17150, 2021.

[4] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage. Federated learning for mobile keyboard prediction, 2019.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[6] A. Krizhevsky, V. Nair, and G. Hinton. *The CIFAR-10 dataset.*, 2009. "`https://www.cs.toronto.edu/~kriz/cifar.html`".

[7] D. Kwon, J. Jeon, S. Park, J. Kim, and S. Cho. Multiagent ddpg-based deep learning for smart ocean federated learning iot networks. *IEEE Internet of Things Journal*, 7(10):9895–9903, 2020.

[8] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

[9] Y. Qu, S. R. Pokhrel, S. Garg, L. Gao, and Y. Xiang. A blockchained federated learning framework for cognitive computing in industry 4.0 networks. *IEEE Transactions on Industrial Informatics*, 17(4):2964–2973, 2021.

[10] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017. "`https://github.com/zalandoresearch/fashion-mnist`".

[11] J. Xu, B. S. Glicksberg, C. Su, P. Walker, J. Bian, and F. Wang. Federated learning for healthcare informatics. *Journal of Healthcare Informatics Research*, 5, 2021.

[12] D. Yin, Y. Chen, R. Kannan, and P. Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5650–5659. PMLR, 10–15 Jul 2018.

[13] L. Zhu, Z. Liu, and S. Han. Deep leakage from gradients. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.