Enhancing IoT Security with Asynchronous Federated Learning for Seismic Inversion

Daniel Manu*, Youzuo Lin†‡, Jingjing Yao§, Zhirun Li*, and Xiang Sun*
*Department of Electrical and Computer Engineering, University of New Mexico, Albuquerque, NM 87131, USA.
†School of Data Science and Society, University of North Carolina at Chapel Hill, Chapel Hill, NC 27599, USA.
‡Los Alamos National Laboratory, Los Alamos, NM, 87545, USA.
§Department of Computer Science, Texas Tech University, Lubbock, TX 79409, USA.

2024 IEEE International Conference on Communications Workshops (ICC Workshops) | 979-8-3503-0405-3/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICCWORKSHOPS59551.2024.10615411

***Abstract*—Seismic data/images are critical to understand the structure of subsurfaces. However, for accurate structural analysis, seismic images need to be converted into velocity images that can recognize the depth and thickness of subsurface layers, and this conversion is usually achieved by using seismic full-waveform inversion (FWI). Various Deep neural networks (DNNs) have recently been proposed to replace FWI, and well-trained DNNs typically have lower computational costs but generate similar velocity images as compared to FWI. Yet, training a DNN model is non-trivial, which requires transmitting seismic data that are sensed by seismic receivers in the field to a centralized data center, leading to data privacy and security issues. Seismic field tests are normally conducted in rural areas, where data centers and Internet infrastructures are not available. Hence, it is impossible to train a DNN and achieve seismic inversion in real-time. In this paper, we propose the Asynchronous Federated Learning for Seismic Inversion (AsyncFedInv) framework, which applies multiple IoT devices in terms of edge computing boards to collaboratively train a compact UNet model in real-time based on a novel asynchronous federated learning, where 1) a staleness function is applied to mitigate model staleness, and 2) clients that generate similar local models would suspend its training, thus reducing the communication costs and energy consumption. Simulation results demonstrate that AsyncFedInv achieves a similar convergence rate but lower training loss and better testing performance as compared to a baseline algorithm FedAvg.**

***Index Terms*—Seismic inversion, federated learning, asynchronous**

I. Introduction

Seismic data/images refer to data collected from an exploration method of sending acoustic waves into the earth and recording the wave reflection and refraction to indicate the type, size, shape, and depth of subsurface layers [1, 2]. As shown in Fig. 1, to collect seismic data, a vibroseis truck is equipped with a seismic source in terms of a hydraulic vibrator to generate controlled and low-frequency vibrations in terms of acoustic waves, which are transmitted into and reflected by the subsurface layers, and finally received by a set of seismic receivers deployed at the ground. However, for accurate structural analysis, an effort should be made to convert seismic waves collected by different receivers over different time slots into depth and thickness of subsurface layers, and this type of conversion is referred to as seismic inversion. Seismic full-waveform inversion (FWI) is one of the most widely used models to extract the Earth's physical characteristics from the seismic data.

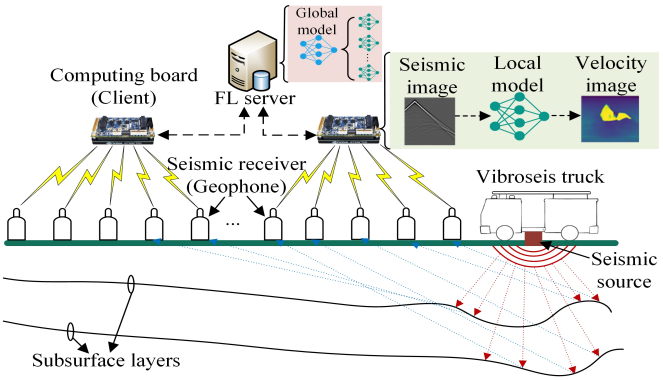


Fig. 1: Federated learning for seismic inversion.

Basically, FWI is to derive the optimal parameters of a forward operator f such that the mean square error (MSE) between recorded seismic data, denoted as $\boldsymbol{d}$, and seismic data generated by the forward operator f is minimized, i.e., $\min ||\boldsymbol{d} - f(\boldsymbol{m})||^2$, where $f(\boldsymbol{m})$ is the simulated seismic data from forward operator f by feeding it with the seismic velocity, denoted as $\boldsymbol{m}$ [3]. The two major drawbacks of FWI include 1) high computational cost, where FWI requires numerous seismic wavefield simulations and reconstructions in each iteration to minimize the MSE, and 2) high sensitivity to noise and errors in the data [4], where small errors in seismic data measurement or initial model can result in large errors in the velocity images generated by FWI.

To resolve the above two drawbacks of FWI, deep neural networks (DNNs) have been proposed and demonstrated their performance in terms of lower computational cost and higher noise resistance [5–7]. Wang and Ma [8] employed a DNN to implement seismic inversion. The DNN comprises a number of fully connected layers followed by a fully convolutional network (FCN). Zhang and Lin [2] utilized a Generative Adversarial Network (GAN) to learn a mapping function that transforms raw seismic waveform into velocity images. Araya-

This work was supported by the National Science Foundation under Award under grant no. CNS-2323050 and CNS-2148178, where CNS-2148178 is supported in part by funds from federal agency and industry partners as specified in the Resilient & Intelligent NextG Systems (RINGS) program.

Authorized licensed use limited to: UNIVERSITY OF NEW MEXICO. Downloaded on April 04,2025 at 01:08:03 UTC from IEEE Xplore. Restrictions apply.

Polo *et al.* [9] introduced a velocity prediction technique using a DNN that takes calculated velocity feature semblance as input, followed by post-processing with a k-means method. Li *et al.* [10] conducted an in-depth analysis on mapping seismic data to velocity images, and developed a novel DNN framework, entitled SeisInvNet, to perform end-to-end velocity inversion mapping, employing enhanced single-trace seismic data as inputs.

All the mentioned DNN models for seismic inversion are trained based on a centralized manner, meaning that seismic data sensed by different seismic receivers have to be transmitted to a data center, which leads to data privacy and security concerns. Also, seismic field tests are normally conducted in rural areas, where facilities, such as data centers and Internet infrastructures, are not available. Hence, these high volumes of seismic data need to be manually transported to a data center for further processing, thus unable to train the DNN model and achieve seismic inversion in real-time. To achieve real-time training and testing of a DNN model for seismic inversion, we incorporate federated learning (FL) [11] and edge computing [12] into seismic inversion. Specifically, as shown in Fig. 1, each seismic receiver is associated with an IoT device in terms of a computing board to upload its seismic data, and the IoT device acts as a client in FL to store and process its received seismic data. Here, FL allows different clients to collaboratively train a machine learning model without sharing their seismic data, thus ensuring IoT data privacy and security. The whole training process is divided into a sequence of global rounds, each composed of four steps. 1) The FL server, which could be one of the IoT devices in the test field, selects suitable clients and broadcasts the current global model to these selected clients. 2) Each selected client independently trains the received global model using its local seismic data to obtain the local model. 3) The selected clients then upload their local models to the FL server via the wireless local network. 4) Once all local models from selected clients have been received, the FL server aggregates them, using, for example, FedAvg [13], to update the global model. The global rounds continue until the global model converges or the number of global rounds exceeds a predefined threshold.

The above FL process is referred to as synchronous FL, where the FL server has to wait until all the selected clients have uploaded their local models, which may lead to the straggler problem if one of the selected clients requires a much longer time to compute and upload its local model, thus increasing the latency of a global round. Although many solutions have been proposed to mitigate the straggler problem, they all raise other issues. For example, one of the most popular solutions is to set up a deadline, and the FL server only selects the clients, who can upload their local models before the deadline, or simply ignores the local models that are uploaded after the deadline [14, 15]. This solution may degrade the global model accuracy caused by biased client selection, i.e., the FL server does not select slow clients, and so the derived global model may not accommodate the training date from these slow clients [16]. Instead of applying synchronous FL, we propose to use asynchronous FL, where the FL server does not wait for all the selected clients to upload their local models; instead, it updates the global model when it receives a local model from any client, and then sends the updated global model to that client. Hence, asynchronous FL does not have the straggler problem and enables all the clients to participate in the training process, thus potentially leading to more efficient training and enhancing data diversity. Many works have demonstrated asynchronous FL outperforms synchronous FL [17], but also highlight two drawbacks of asynchronous FL. 1) Model staleness, where slow clients train their local models based on outdated global models; hence, the global model may reduce its accuracy when the FL server aggregates the local models from those slow clients [17]. 2) High communication cost, where fast clients need to communicate with the FL server much more frequently, thus leading to high energy consumption.

In this paper, we propose the Asynchronous Federated Learning for Seismic Inversion (AsyncFedInv) framework, which integrates asynchronous FL and edge computing into seismic data analysis to achieve real-time seismic inversion. The main contributions are summarized as follows.

- We propose AsyncFedInv that enables edge devices to apply asynchronous FL to train a DNN, i.e., UNet, to achieve real-time seismic inversion. To the best of our knowledge, we are the first that explore the integration of FL and edge computing into seismic inversion.
- In AsyncFedInv, to efficiently train UNet based on asynchronous FL, we apply 1) staleness functions to dynamically adjust the weights of local models during model aggregation, thus potentially mitigating the model staleness issue, and 2) dynamic client participation, i.e., the clients, whose local models do not have a significant difference from the current global model, will suspend its training, thus reducing the communication costs and energy consumption, while preserving the model accuracy.
- Extensive simulations have been conducted to demonstrate the convergence and model accuracy of AsyncFedInv. Also, the performance of AsyncFedInv by applying different staleness functions is evaluated via simulations.

II. Data-Driven FWI (DD-FWI) Design

DD-FWI is to derive a pseudoinverse operator $F_{\boldsymbol{\theta}}^{\dagger} : \boldsymbol{R}^D \rightarrow \boldsymbol{R}^M$ parameterized by $\boldsymbol{\theta}$, where

$$F_{\boldsymbol{\theta}}^{\dagger}(\boldsymbol{P}^{\text{obs}}) \approx \boldsymbol{m}^{\text{true}} + \varepsilon, \tag{1}$$

where $\boldsymbol{P}^{\text{obs}} \in \boldsymbol{R}^D$ is the seismic data observed by different seismic receivers, $\boldsymbol{m}^{\text{true}} \in \boldsymbol{R}^M$ is the ground truth of the velocity data, and ε is the measurement noise. In 2-D scenarios, the dimensions of $\boldsymbol{P}^{\text{obs}}$ and $\boldsymbol{m}^{\text{true}}$ are defined as $D = S \times R \times T$ and $M = H \times W$, respectively. Here, S R, and T represent the number of seismic sources generated in a period, the number of seismic receivers, and timesteps in a period for inversion, respectively. Also, $H \times W$ is the size

Authorized licensed use limited to: UNIVERSITY OF NEW MEXICO. Downloaded on April 04,2025 at 01:08:03 UTC from IEEE Xplore. Restrictions apply.

of a velocity image, which typically equals $R \times T$. In DD-FWI, $F_{\boldsymbol{\theta}}^{\dagger}$ is normally implemented as a DNN, such as UNet [18, 19], which learns a mapping from the seismic data to the velocity data. Basically, the parameter $\boldsymbol{\theta}$ in $F_{\boldsymbol{\theta}}^{\dagger}$ is trained based on, for example, stochastic gradient descent, given a set of seismic data $\left\{\boldsymbol{P}_1^{obs}, \cdots, \boldsymbol{P}_N^{obs}\right\}$ and their corresponding velocity images $\{\boldsymbol{m}_1^{true}, \cdots, \boldsymbol{m}_N^{true}\}$, where velocity images can be obtained based on, for example, the physics-based FWI model, which has high computational cost but provides accurate velocity images. The loss function of UNet is

$$\underset{\boldsymbol{\theta}}{\operatorname{argmin}}\ \mathcal{L} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^{N} \|\boldsymbol{m}_i^{true} - F_{\boldsymbol{\theta}}(\boldsymbol{P}_i^{obs})\|^2. \quad (2)$$

UNet is based on an encoder-decoder structure, where the encoder reduces the dimensions of the inputs by extracting relevant high-level features, and the decoder transforms the extracted features into a different domain. We modify our previous UNet model [20] to achieve seismic inversion, and the new UNet model has a more compact architecture to be executed in resource-constrained clients. As shown in Fig. 2, the encoder has four convolutional layers with 3×3 fixed kernel size and both stride and padding equal to 1. Each convolutional layer is then followed by a 2×2 max-pooling with the stride of 2 to downsample the feature maps. At each downsampling step, the number of feature channels is doubled. The channel dimensions in the encoding are 32, 64, 128, and 256. The decoder comprises four transposed convolutional layers for upsampling, each with 5×5 kernel size, stride of 2, and zero padding. At each upsampling step, the number of feature channels is halved. The channel dimensions in the decoding path are 512, 256, 128, 64, and 32. Finally, a 3×3 convolutional layer maps the last 32 features into the predicted velocity value. The soft-max function is then used to obtain the predicted velocity label. Basically, our modified UNet architecture has a total of 9 convolutional layers.

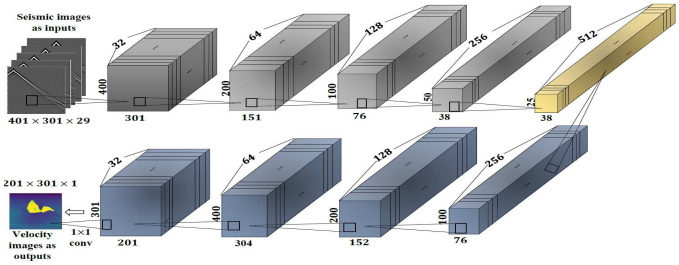


Fig. 2: The designed UNet model for seismic inversion.

III. System Models

The clients apply FL to collaboratively train the UNet model such that the global loss function $\mathcal{L}$ can be minimized, i.e.,

$$\underset{\boldsymbol{\theta}}{\operatorname{argmin}}\ \mathcal{L} = \underset{\boldsymbol{\theta}}{\arg\min} \sum_{k \in \boldsymbol{\mathcal{K}}} \frac{N_k}{N} \mathcal{L}_k(\boldsymbol{\theta}), \quad (3)$$

where $\boldsymbol{\theta}$ is the set of global model parameters, $\boldsymbol{\mathcal{K}}$ is the set of clients, N_k is the number of seismic data and velocity image pairs, i.e., $\left\{\boldsymbol{P}_i^{obs}, \boldsymbol{m}_i^{true}\right\}$ in client k, $N = \sum_{k \in \boldsymbol{\mathcal{K}}} N_k$ is the total number of seismic data and velocity image pairs for all the clients, and $\mathcal{L}_k(\boldsymbol{\omega})$ is the local loss function of client k, i.e.,

$$\mathcal{L}_k(\boldsymbol{\theta}) = \frac{1}{N_k} \sum_{n=1}^{N_k} \mathcal{L}\left(\boldsymbol{\theta}, \boldsymbol{P}_n^{obs}, \boldsymbol{m}_n^{true}\right). \quad (4)$$

Since Problem (3) is nontrivial to solve, FL decomposes the problem to enable each client to derive its local model $\boldsymbol{\theta}_k$ based on its local data set to minimize the loss function, i.e.,

$$\begin{aligned} &\arg\min_{\boldsymbol{\theta}_k} \frac{1}{N_k} \sum_{n=1}^{N_k} \mathcal{L}\left(\boldsymbol{\theta}_k, \boldsymbol{P}_n^{obs}, \boldsymbol{m}_n^{true}\right) \quad (5) \\ &= \arg\min_{\boldsymbol{\theta}_k} \frac{1}{N_k} \sum_{n=1}^{N_k} \|\boldsymbol{m}_n^{true} - F_{\boldsymbol{\theta}_k}(\boldsymbol{P}_n^{obs})\|^2, \end{aligned}$$

and then local models are aggregated at the FL server to derive a global model.

A. Latency Models

In this section, we present the latency of a client, which will be used to evaluate the training/convergence time of FL algorithms. In general, the latency of client k comprises 1) the latency of client k in downloading the global model from the FL server, defined as $t_k^{download}$, 2) the latency of client k to compute its local model, defined as t_k^{comp}, 3) the latency of client k when uploading its local model to the FL server, defined as t_k^{upload}, and 4) latency of the FL server to update the global model. Typically, the latency related to the global model update in Step 4) is negligible. Thus, we define the latency of client k in each global iteration as follows:

$$t_k = t_k^{download} + t_k^{comp} + t_k^{upload}. \quad (6)$$

1) Computing Latency: Assume that there are N_k seismic images used by client k to train its local UNet model. The batch size is η, and so the number of batches for client k is $\frac{N_k}{\eta}$. Also, assume that the number of local iterations of training a local model is ϕ. Hence, there are $N_k \times \phi$ and $\frac{\phi N_k}{\eta}$ number of forward and backward propagations, respectively, in client k's global round. If the complexity in terms of the number of CPU cycles required to conduct a forward and backward propagation for the proposed UNet model in client k are $c_k^{forward}$ and $c_k^{backward}$, respectively, then the computing latency of client k to train the local model is

$$t_k^{comp} = \frac{N_k \times \phi \times c_k^{forward} + \frac{\phi N_k}{\eta} \times c_k^{backward}}{f_k}, \quad (7)$$

where f_k is the CPU frequency of client k.

2) Downloading and Uploading Latency: Assume that a time division duplex system is applied for the FL server to download the global model to a client as well as for a client to upload a local model to the FL server. Denote g_k as the channel gain between the FL server to client k, and the channel

Authorized licensed use limited to: UNIVERSITY OF NEW MEXICO. Downloaded on April 04,2025 at 01:08:03 UTC from IEEE Xplore. Restrictions apply.

is reciprocal. Hence, the downloading and uploading data rate achievable by client k are

$$\begin{cases} r_k^{downlload} = B\log_2(1+\frac{p^s g_k}{N_0}), \\ r_k^{upload} = B\log_2(1+\frac{p_k^c g_k}{N_0}), \end{cases} \tag{8}$$

Here, B is the total amount of bandwidth, p^s and p_k^c are the transmission power density of the FL server and client k, respectively, and N_0 is the average noise power density. Assuming that the size of the global model is s, then we have

$$\begin{cases} t_k^{downlload} = \frac{s}{r_k^{downlload}} = \frac{s}{B\log_2(1+\frac{p^s g_k}{N_0})}. \\ t_k^{upload} = \frac{s}{r_k^{upload}} = \frac{s}{B\log_2(1+\frac{p_k^c g_k}{N_0})}. \end{cases} \tag{9}$$

IV. The AsyncFedInv Framework

AsyncFedInv modifies the existing asynchronous FL to achieve more efficient training of the proposed UNet model. As mentioned before, asynchronous FL is different from synchronous FL since the FL server immediately updates the global model $\boldsymbol{\theta}^{global}$ upon receiving a local model, i.e.,

$$\boldsymbol{\theta}^{global} := (1-\alpha_k)\boldsymbol{\theta}^{global} + \alpha_k \boldsymbol{\theta}_k^{local}, \tag{10}$$

where α_k is the weight assigned to client k. As mentioned before, the staleness problem is one of the major drawbacks of asynchronous FL, where some slow clients train their local models $\boldsymbol{\theta}_k^{local}$ based on the outdated global model. As a result, these local models may not improve the performance of the global model, or even diverge the global model, when they are used to update the global model based on Eq. (10). One common approach to address staleness is to adjust α_k. That is, a slow client, which has a higher latency t_k to compute and upload its local model, will be assigned with a smaller weight α_k, and vice versa. Specifically, α_k is calculated based on

$$\alpha_k = \alpha \times h(t_k), \tag{11}$$

where α ($0 \leq \alpha \leq 1$) is a hyperparameter and $h(t_k)$ is client $k's$ staleness function to evaluate the freshness of the local model. Here, we design three typical staleness functions, i.e.,

- **Constant:** $h^{const} = 1$.
- **Exponential:** $h^{exp}(t_k) = e^{-a(t_k-\tau)}$.
- **Hinge:** $h^{hinge}(t_k) = \begin{cases} 1, & \text{if } t_k - \tau \leq b, \\ \frac{1}{a(t_k-\tau-b)+1}, & \text{otherwise}, \end{cases}$

where a and b are hyperparameters, and $\tau = \min(t_k | \forall k \in \mathcal{K}')$. Here, $\mathcal{K}' \subset \mathcal{K}$ is the set of clients, who are currently training their local models. Basically, h^{const} treats all the clients equally in terms of staleness/weights, and $h^{exp}(t_k)$ and $h^{hinge}(t_k)$ assigns lower weights to the clients with higher t_k.

Another drawback of asynchronous FL is the fast clients' high communication and computing costs. That is, the fast clients have to frequently upload and keep training their local models, thus leading to high energy consumption. Note that energy consumption could be critical to the clients, who are deployed in the rural field and powered by portable batteries. To reduce the cost, we apply a simple but effective solution, which is to suspend a client in training its local model by not sending the updated global model if the client's uploaded local model is similar to the updated global model, i.e.,

$$\left\| \boldsymbol{\theta}^{global} - \boldsymbol{\theta}_k^{local} \right\| \leqslant \varepsilon, \tag{12}$$

where ε is the predefined threshold. Once the FL server updates the global model, it will evaluate all the clients in $\mathcal{K}''$ to see if Eq. (12) is still met, where $\mathcal{K}''$ is the set of clients who are suspending their training, and $\mathcal{K}'' \cup \mathcal{K}' = \mathcal{K}$. If a client in $\mathcal{K}''$ does not satisfy Eq. (12), the FL server will resume its training by sending the current global model. Algorithm 1 summarizes AsyncFedInv.

Algorithm 1: The AsyncFedInv framework

```
Initialize θ^global and hyperparameters.
Broadcast θ^global to all the clients.
At the FL server:
while receive a local model θ_k^local from client k do
    Calculate α_k based on Eq. (11);
    Update θ^global based on Eq. (10);
    if ||θ^global − θ_k^local|| > ε then
        Download θ^global to client k;
    else
        K' := K'\k and K'' := K'' ∪ k;
        Store client k's local model θ_k^local ;
    end
    for each k' ∈ K'' do
        if ||θ^global − θ_k'^local|| > ε then
            K' := K' ∪ k' and K'' := K''\k';
            Download θ^global to client k';
        end
    end
end
At client k:
while receive a global model θ^global do
    θ_k^local = θ^global;
    Train θ_k^local based on batch gradient descent;
    Upload θ_k^local to the FL server;
end
```

V. Simulations

A. Simulation parameters

Assume that there are 3 clients to receive data from seismic receivers, and each client trains the UNet model described in Fig. 2 by using the Adam optimizer with a learning rate of 0.001. The 2D SEG Salt data set [18] is used, where each seismic image represents the pressure of the acoustic waves received by 301 receivers during the 2-second monitoring period. These 301 receivers are evenly distributed in a 3 km distance area and each receiver samples the acoustic waves 201 times during the 2-second monitoring period once the transmitter generates an acoustic wave. Hence, the size of each seismic image is 301 $\times$ 201 pixels, and Fig. 3 (left) provides an example of a seismic image. In the data set, there

Authorized licensed use limited to: UNIVERSITY OF NEW MEXICO. Downloaded on April 04,2025 at 01:08:03 UTC from IEEE Xplore. Restrictions apply.

are 29 sources placed at different locations in a field. The total number of seismic images in the data set is 120. Note that the data set does not contain the ground truth in terms of the velocity images of the corresponding seismic images. Since FWI has been demonstrated to generate high accurate velocity images based on the low-noisy seismic images, such as 2D SEG Salt data set, we consider the velocity images derived by the FWI model in [21] as the ground truth values. Note that each velocity image is assumed to consist of 5 to 12 layers, representing the background velocity. The velocity value associated with each pixel in a velocity image varies between 2,000 and 4,000 m/s and the pixel that represents the salt has the velocity value of 4,500 m/s. The size of each velocity image should be matched up with the size of its seismic image, which is 301 $\times$ 201 pixels. Fig. 3 (right) shows a velocity image that is generated by FWI based on the seismic image on the left.

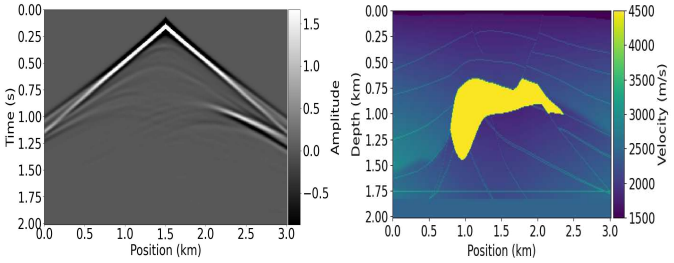


Fig. 3: An example of a pair of seismic (left) and velocity (right) images in the 2D SEG Salt dataset, where the velocity image is generated by FWI.

TABLE I: Simulation parameters

Parameter	Value
Size of the local model s	100 kbit
Number of local iterations ϕ	50
Number of global iterations	100
Batch size η	5
Hyperparameters in staleness functions a/b	10/4
Hyperparameter α in Eq. (10)	0.5

All 120 pairs of seismic/velocity images are first grouped into different classes based on the similarity in terms of structural similarity index metric (SSIM) of two velocity images from two pairs of seismic/velocity images. That is, if the SSIM value between a velocity image and any velocity image in a class is no larger than the defined threshold ξ, the pair of seismic/velocity images will be grouped into that class. Based on that, the 120 seismic/velocity image pairs are then distributed to the 3 clients in two different settings, i.e., independent and identically distribution (IID) and non-IID. In IID, the clients have the same/similar number of image pairs in a class. In non-IID, the probability of having x_l pairs of images in class l at client k is assumed to follow a Dirichlet distribution [22], i.e.,$f(x_1, x_2, \ldots, x_L; \beta) = \frac{\Gamma(\beta L)}{\Gamma(\beta)^L} \prod_{l=1}^{L} x_l^{\beta-1}$, where L is the total number of classes, $\Gamma()$ is the gamma function, $\Gamma(z) = \int_0^\infty y^{z-1} e^{-y} dy$ and β is the concentration parameter that determines the level of label imbalance. A larger β results in a more balanced data partition among different labels within a client (i.e., closer to IID). Other simulation parameters are listed in Table I.

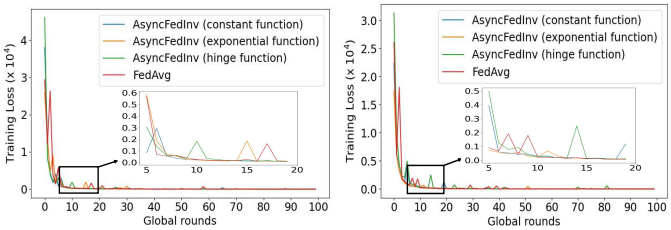


Fig. 4: Loss curves for AsyncFedInv and FedAvg in IID (left) and non-IID $\beta = 0.1$ (right) settings.

B. Simulation results

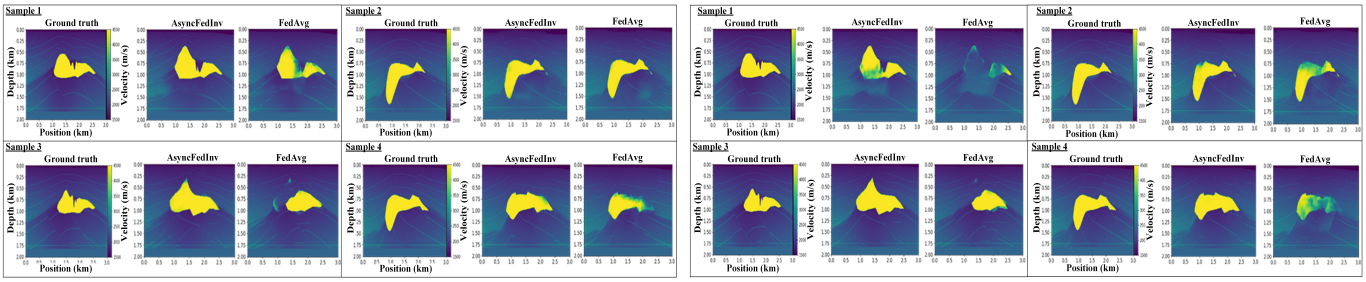
We compare the performance of AsyncFedInv with a baseline solution, i.e., FedAvg, which a well-known synchronous FL method to invite all the clients to participate in the model training in each global iteration.

1) Training loss and convergence analysis: Fig. 4 shows the loss curves learning curves of FedAvg and AsyncFedInv with different staleness functions under IID and non-IID cases. Note that there is no definition of a global round in asynchronous FL. To facilitate the comparisons, we define the duration of a global round in both AsyncFedInv and FedAvg as the duration of the slowest client in computing and uploading its local model. From Fig. 4, we can observe that all the methods finally converge with a very similar rate in both IID and non-IID cases. Yet, AsyncFedInv with exponential staleness function achieves the lowest training loss when the curve is converged. Specifically, the loss of the FedAvg, AsyncFedInv with exponential, AsyncFedInv with hinge, and AsyncFedInv with constant are 3.1449, 2.2356, 2.6344, and 3.2033, respectively, in IID, and 4.4193, 1.7669, 3.1383, 6.4552, respectively, in non-IID. Hence, we will use the exponential staleness function as the default setting for AsyncFedInv later on.

2) Testing result analysis: By feeding the testing data sets into the UNet models that are generated by AsyncFedInv and FedAvg after 100 global rounds, the testing results are obtained. Fig. 5 show four typical samples in IID and non-IID cases, respectively, where each sample comprises three different velocity images inverted from the same seismic image, i.e., a ground truth velocity image generated by FWI, a velocity image generated by AsyncFedInv, and a velocity image generated by FedAvg. From the figures, we can clearly identify that the velocity images generated by AsyncFedInv are more similar to the ground truth than FedAvg in IID. For instance, the velocity image generated by AsyncFedInv in the first sample is much better than that generated by FedAvg. Also, the testing result of FedAvg becomes much worse than AsyncFedInv in the non-IID case. For instance, the velocity image generated by FedAvg in the first sample is totally different from the ground truth, while the velocity image generated by AsyncFedInv still keeps high similarity.

In order to quantify the testing results, two metrics, i.e., structural similarity index metric (SSIM) and peak signal-to-

Authorized licensed use limited to: UNIVERSITY OF NEW MEXICO. Downloaded on April 04,2025 at 01:08:03 UTC from IEEE Xplore. Restrictions apply.



(a) IID (b) Non- IID

Fig. 5: Testing result for AsyncFedInv and FedAvg in IID (left) and non-IID (right).

TABLE II: SSIM and PSNR for AsyncFedInv and FedAvg

	IID				Non-IID			
	SSIM		PSNR		SSIM		PSNR	
Sample	Ours	FedAvg	Ours	FedAvg	Ours	FedAvg	Ours	FedAvg
0	0.541	0.508	15.371	15.900	0.532	0.502	15.044	12.168
1	0.591	0.612	17.801	17.969	0.664	0.601	18.438	18.508
2	0.537	0.528	12.441	14.517	0.516	0.527	12.227	14.780
3	0.568	0.557	25.278	24.682	0.596	0.553	25.919	24.312
4	0.448	0.424	17.713	15.016	0.433	0.410	18.200	15.873
5	0.609	0.543	14.924	15.250	0.607	0.503	14.770	15.038
6	0.476	0.429	19.640	12.159	0.471	0.470	20.125	15.229
7	0.682	0.659	16.218	13.843	0.635	0.648	14.115	17.273
Average	**0.557**	0.533	**17.424**	16.167	**0.557**	0.527	**17.355**	16.648

noise ratio (PSNR), are used. SSIM and PSNR evaluate the similarity and quality of a predicted velocity image, respectively, in comparison to its ground truth. A higher SSIM/PSNR value indicates a higher similarity/quality of the predicted velocity image. Table II shows testing results on 8 seismic images for AsyncFedInv and FedAvg in IID and non-IID cases, where we can observe that AsyncFedInv always incurs lower average SSIM and PSNR values than FedAvg in both IID and non-IID cases. All in all, we conclude that AsyncFedInv has a similar convergence rate, but lower training loss and better testing performance as compared to FedAvg.

VI. Conclusions

In this paper, we have proposed AsyncFedInv that uses multiple edge devices to collaboratively train a compact UNet model in real-time based on a novel asynchronous federated learning, where 1) a staleness function is used to mitigate model staleness, and 2) dynamic client participation is designed to reduce the communication costs and energy consumption of clients. Simulation results show that AsyncFedInv achieves a similar convergence rate but lower training loss and better testing performance as compared to FedAvg.

References

[1] J. Virieux and S. Operto, "An overview of full-waveform inversion in exploration geophysics," *Geophysics*, vol. 74, no. 6, pp. WCC1–WCC26, 2009.

[2] Z. Zhang and Y. Lin, "Data-driven seismic waveform inversion: A study on the robustness and generalization," *IEEE Transactions on Geoscience and Remote sensing*, vol. 58, no. 10, pp. 6900–6913, 2020.

[3] Y. Ma, D. Hale, B. Gong, and Z. Meng, "Image-guided sparse-model full waveform inversion," *Geophysics*, vol. 77, no. 4, pp. R189–R198, 2012.

[4] A. Fichtner, *Full seismic waveform modelling and inversion*. Springer Science & Business Media, 2010.

[5] J. Kim, J. K. Lee, and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks," in *IEEE conference on computer vision and pattern recognition*, 2016, pp. 1646–1654.

[6] F. Liu, C. Shen, and G. Lin, "Deep convolutional neural fields for depth estimation from a single image," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 5162–5170.

[7] Y. Wu and Y. Lin, "Inversionnet: An efficient and accurate data-driven full waveform inversion," *IEEE Transactions on Computational Imaging*, vol. 6, pp. 419–433, 2019.

[8] W. Wang and J. Ma, "Velocity model building in a crosswell acquisition geometry with image-trained artificial neural networks," *Geophysics*, vol. 85, no. 2, pp. U31–U46, 2020.

[9] M. Araya-Polo, J. Jennings, A. Adler, and T. Dahlke, "Deep-learning tomography," *The Leading Edge*, vol. 37, no. 1, pp. 58–66, 2018.

[10] S. Li, B. Liu, Y. Ren, Y. Chen, S. Yang, Y. Wang, and P. Jiang, "Deep-learning inversion of seismic data," *arXiv preprint arXiv:1901.07733*, 2019.

[11] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[12] X. Sun and N. Ansari, "Edgeiot: Mobile edge computing for the internet of things," *IEEE Communications Magazine*, vol. 54, no. 12, pp. 22–29, 2016.

[13] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*, 2017, pp. 1273–1282.

[14] L. Yu, R. Albelaihi, X. Sun, N. Ansari, and M. Devetsikiotis, "Jointly optimizing client selection and resource management in wireless federated learning for internet of things," *IEEE Internet of Things Journal*, vol. 9, no. 6, pp. 4385–4395, 2022.

[15] R. Albelaihi, A. Alasandagutti, L. Yu, J. Yao, and X. Sun, "Deep-reinforcement-learning-assisted client selection in nonorthogonal-multiple-access-based federated learning," *IEEE Internet of Things Journal*, vol. 10, no. 17, pp. 15 515–15 525, 2023.

[16] L. Yu, X. Sun, R. Albelaihi, and C. Yi, "Latency aware semi-synchronous client selection and model aggregation for wireless federated learning," *arXiv preprint arXiv:2210.10311*, 2022.

[17] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," *CoRR*, vol. abs/1903.03934, 2019. [Online]. Available: http://arxiv.org/abs/1903.03934

[18] F. Yang and J. Ma, "Deep-learning inversion: A next-generation seismic velocity model building method," *Geophysics*, vol. 84, no. 4, pp. R583–R599, 2019.

[19] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer, 2015, pp. 234–241.

[20] D. Manu, P. M. Tshakwanda, Y. Lin, W. Jiang, and L. Yang, "Seismic waveform inversion capability on resource-constrained edge devices," *Journal of Imaging*, vol. 8, no. 12, p. 312, 2022.

[21] A. Guitton, G. Ayeni, and E. Díaz, "Constrained full-waveform inversion by model reparameterization," *Geophysics*, vol. 77, no. 2, pp. R117–R127, 2012.

[22] A. A. Gouda and T. Szántai, "On numerical calculation of probabilities according to dirichlet distribution," *Annals of Operations Research*, vol. 177, pp. 185–200, 2010.

Authorized licensed use limited to: UNIVERSITY OF NEW MEXICO. Downloaded on April 04,2025 at 01:08:03 UTC from IEEE Xplore. Restrictions apply.