Communication-Efficient Federated Learning: A Second Order Newton-Type Method With Analog Over-the-Air Aggregation

Mounssif Krouka, Anis Elgabli, *Member, IEEE*, Chaouki Ben Issaid, *Member, IEEE*, and Mehdi Bennis, *Fellow, IEEE*

(Invited Paper)

***Abstract*—Owing to their fast convergence, second-order Newton-type learning methods have recently received attention in the federated learning (FL) setting. However, current solutions are based on communicating the Hessian matrices from the devices to the parameter server, at every iteration, incurring a large number of communication rounds; calling for novel communication-efficient Newton-type learning methods. In this article, we propose a novel second-order Newton-type method that, similarly to its first-order counterpart, requires every device to share only a model-sized vector at each iteration while hiding the gradient and Hessian information. In doing so, the proposed approach is significantly more communication-efficient and privacy-preserving. Furthermore, by leveraging the over-the-air aggregation principle, our method inherits privacy guarantees and obtains much higher communication efficiency gains. In particular, we formulate the problem of learning the *inverse Hessian-gradient product* as a quadratic problem that is solved in a distributed way. The framework alternates between updating the *inverse Hessian-gradient product* using a few alternating direction method of multipliers (ADMM) steps, and updating the global model using Newton's method. Numerical results show that our proposed approach is more communication-efficient and scalable under noisy channels for different scenarios and across multiple datasets.**

***Index Terms*—Distributed optimization, communication-efficient federated learning, second-order methods, analog-over-the-air aggregation, ADMM.**

I. Introduction

RECENT developments in 5G networks and Internet of Things (IoT) applications led to the pervasiveness of connectivity of many devices and sensors generating massive amounts of data. This influx of data is used to train machine learning (ML) algorithms that span different applications, such as trajectory optimization [1], image classification [2], and wireless resource scheduling [3]. Traditionally, training such ML algorithms is done in a centralized way in which multiple devices send their raw and often-times private data to a centralized parameter server (PS) equipped with high computational capabilities for training purposes. However, the available data for training such ML algorithms is generated across several devices that, in many cases, require privacy-preserving mechanisms to ensure data integrity. Moreover, the bandwidth limitations create a bottleneck when transferring a huge amount of data from devices to the PS [4]. For this reason, federated and distributed learning algorithms have received significant attention owing to the fact that these ML models can be trained in a totally distributed manner without sharing raw and private data.

Manuscript received 29 January 2022; revised 25 March 2022; accepted 28 April 2022. Date of publication 9 May 2022; date of current version 19 August 2022. This work was supported in part by the Academy of Finland 6G Flagship under Grant 318927; in part by Project SMARTER; in part by Projects EU-ICT IntellIoT and EUCHISTERA LearningEdge; and in part by CONNECT, Infotech-NOOR, and NEGEIN. The editor coordinating the review of this article was Z. Niu. *(Corresponding author: Mounssif Krouka.)*

The authors are with the Center of Wireless Communications, University of Oulu, 90014 Oulu, Finland (e-mail: mounssif.krouka@oulu.fi; anis.elgabli@oulu.fi; chaouki.benissaid@oulu.fi; mehdi.bennis@oulu.fi).

Digital Object Identifier 10.1109/TGCN.2022.3173420

A. Related Works

Several papers have addressed communication-efficient solutions for solving distributed learning problems. The proposed works use different techniques and approaches based on first-order and second-order methods, as discussed next.

1) First-Order Methods: Typically, distributed model training is based on using first-order methods, such as federated learning (FL) [5], distillation (FD) [6], and many others. With distributed first-order methods, the model update requires a large number of uplink and downlink exchanges, in terms of communication rounds, between the devices and the PS until convergence, in addition to substantial energy and bandwidth resources per communication round. This bottleneck can be tackled using two approaches: (i) reducing the number of communication rounds and (ii) minimizing the communication resources per round. Reducing the number of communication rounds can be achieved by accelerating the convergence rate using variance reduction techniques and momentum acceleration [7]–[9]. On the other hand, to reduce communication overhead per round, many techniques have been proposed in the literature, such as gradient reuse [10], [11], quantization [12]–[17], and sparsification [18]–[21].

2) Second-Order Methods: First-order methods suffer from the dependency on the *condition number* [22]. Specifically, for a μ-strongly convex and L-smooth function f, the gradient descent (GD) algorithm can achieve a global linear convergence rate of $1-\frac{2}{\kappa+1}$, where $\kappa=\frac{L}{\mu}$ denotes the *condition number*, for a step-size $\alpha=\frac{2}{\mu+L}$ [23]. Hence, large values of the *condition number* slow the convergence of the

learning algorithms. To overcome the large number of required communication rounds, second-order methods have recently gained considerable attention since they exploit second-order curvature properties that are independent of the *condition number* [22]. Although the second-order Newton-type method improves the iteration complexity in terms of number of communication rounds to reach a target accuracy, it still requires the computation and communication of the Hessian matrix from every device and at every round. This results in a huge $\mathcal{O}(d^2)$ overhead, compared to $\mathcal{O}(d)$ as in first-order methods, where d being the dimension of the trained model. Moreover, the Hessian matrix contains valuable information about the properties of the loss function such as its local curvature, and when shared with the PS, may violate privacy by eavesdroppers or the honest-but-curious PS. For instance, in the medical domain, the authors in [24] used eigenvalues from the Hessian matrix to extract some important information from the input images. Consequently, sharing second-order information violates the privacy of the patients' healthcare data.

Many works have proposed communication-efficient second-order solutions to overcome the problem of communicating the complete Hessian matrix. Works such as [25]–[27] propose methods that rely on quasi-Newton methods, such as the Limited-memory Broyden Fletcher Goldfarb Shanno (L-BFGS), to update the global approximated inverse Hessian matrix at the PS. However, these methods suffer from high computation and communication overhead to achieve a good approximation of the inverse Hessian matrix. Alternatively, other works propose to solve a second-order approximation problem locally at each device, without maintaining and computing a global Hessian matrix [28]–[30]. Nonetheless, a high precision solution to the local problem is required at each device to ensure a good convergence rate. Recently, compression schemes have been introduced for distributed second-order methods to reduce the communication payload size. Authors in [31] propose a communication-efficient Hessian learning technique where a compressed version of the Hessian is transmitted to the PS.

3) Analog Over-the-Air Aggregation: Motivated by the superposition property of signals in the wireless multiple-access channel (MAC), analog over-the-air aggregation is a promising technique that is widely used in first-order methods due to the fact that the PS only requires the sum of the local gradients/models [32]–[35]. However, for existing second-order methods, such an aggregation step is no longer possible since the update at the PS is no longer a simple aggregation step. Instead, it requires decoding the (approximate) Hessian matrices and gradients, inverting the Hessian matrix and then performing multiplications. Even with existing communication-efficient algorithms [29], [30], which involve sending a compressed version of the Hessian, devices have to transmit more than one vector that need to be decoded separately at the PS, highlighting the fact that direct aggregation of the received signals at the PS cannot be applied. Hence, to benefit from the analog over-the-air aggregation savings in terms of communication costs, and the convergence speed of second-order methods, we propose a novel second-order learning technique that requires each device to send a vector of the same model size that is aggregated at the PS (details will be explained in Section III).

B. Contributions and Structure

In this article, we propose a novel second-order learning approach that achieves the fast convergence property of second-order methods and low per iteration communication cost of first-order methods. In particular, the problem of learning the *inverse Hessian-gradient product* is formulated as a quadratic problem and solved in a distributed manner. Namely, we approximate the solution of this problem with a few alternating direction method of multipliers (ADMM) steps where every device communicates only a model-sized vector to the PS, while both the Hessian matrix and the gradient information are concealed. Consequently, the proposed approach saves communication resources and guarantees information privacy. Subsequently, the global model is updated using the Newton's method. Moreover, we leverage analog over-the-air aggregation for faster and more communication-efficient model training. Our major contributions are summarized as follows.

- To the best of our knowledge, this is the first work to leverage the analog over-the-air aggregation strategy for second-order methods in the FL setting. Unlike existing second-order methods that require sharing a significant amount of information, in our proposed algorithm, every device shares a single vector as in first-order methods.
- Within our framework, referred to as *NAAM*–**N**ewton **A**nalog **ADMM**, we propose two variants and their implementations, namely: (i) with channel-inversion (*NAAM-v0*), and (ii) without channel-inversion (*NAAM-v1*). The proposed algorithms extend the existing analog-over-the-air aggregation algorithms in [34], [35], which were proposed for federated learning (FL) with first-order methods.
- The proposed framework ensures privacy since devices neither transmit the Hessian nor the gradient to the PS. Moreover, *NAAM-v1* algorithm provides an additional layer of privacy guarantee by natively incorporating channel perturbations into the problem formulation.
- Numerical results show that our proposed algorithms cope with noisy and time-varying channels and outperform the digital baselines in terms of required communication resources at different signal-to-noise ratio (SNR) regimes. Results also show that for bandwidth-limited systems, *NAAM-v0* and *NAAM-v1* demonstrate a robust performance, especially when the number of devices becomes large. Hence, the proposed framework is a promising scalable second-order methods for FL since the required communication resources are independent of the number of contributing devices.

The rest of the paper is structured as follows. In Section II, we briefly discuss existing first and second-order solutions, in addition to our proposed algorithm where we first describe the main idea under the digital transmission setting. In Section III, we extend our discussion to the analog communication scenario in the presence of time-varying channels. We then show how existing analog-over-the-air aggregation algorithms

TABLE I
TABLE OF NOTATIONS

Notation	Definition
$f_n(\boldsymbol{x})$	local loss function at device n
$\boldsymbol{x}^r$	Model at Newton iteration r
K	Number of ADMM steps
R	Number of Newton iterations
d	Dimension of the model $\boldsymbol{x}$
N	Number of devices
M	Number of samples per device
$\nabla f_i(\boldsymbol{x}^r)$	The gradient vector of $f(\boldsymbol{x}^r)$
$\nabla^2 f_i(\boldsymbol{x}^r)$	The Hessian matrix of $f(\boldsymbol{x}^r)$
$\boldsymbol{H}_n^0$	The Hessian matrix of device n at $r=0$
$\boldsymbol{g}_n^r$	The gradient vector of device n
$\boldsymbol{w}^{r,k+1}$	ADMM update vector at the PS
$\boldsymbol{w}_n^{r,k+1}$	ADMM local version update at device n
$\boldsymbol{\lambda}_n^{r,k+1}$	The dual variable at device n
ρ	Augmented Lagrangian mismatch constant
α	Learning rate
β	Channel inverse threshold
$\hat{z}_i^{r,k+1}$	AWGN noise at subcarrier i
$c_n(t)$	Scaling factor for device n at time instance t
P_n	Power budget of device n
T	Signal receiving duration in seconds
N_0	AWGN power spectral density
Ns	Number of available subcarriers
τ_n	Required uploading time slots for device n

can be further leveraged in our framework and present two implementations, namely *NAAM-v0* with channel inversion and *NAAM-v1* without channel inversion. In Section IV, the effectiveness of our proposed approach is corroborated via simulations. Finally, we conclude our work in Section V.

Notation: $\mathbb{R}$ denotes the set of real numbers whereas the symbols $(.)^*$, $(.)^T$, and $\circ$ are used for conjugate, transpose and Hadamard product operations, respectively. Uppercase-bold letters denote matrices, lowercase-bold letters are reserved for vectors, and scalar values are denoted by regular-font letters. Detailed description of the notations is found in Table I.

II. From First-Order to Second-Order Type Learning Methods

To alleviate the communication bottleneck related to large data and model sizes in modern ML systems, distributed learning is a promising technique that improves performance while scaling to larger input data and number of training devices. In this section, we first introduce first and second-order learning algorithms, then based on their key concepts, we describe our main approach and its key novelty. We consider solving the following FL problem:

$$\min_{\boldsymbol{x}\in\mathbb{R}^d} \; f(\boldsymbol{x}) \;\; \text{with} \;\; f(\boldsymbol{x}) = \frac{1}{N}\sum_{n=1}^{N} f_n(\boldsymbol{x}), \tag{1}$$

where $\boldsymbol{x}$ is the model with dimension d to be trained, N is the number of devices, and $f_n : \mathbb{R}^d \longrightarrow \mathbb{R}$ is the local loss function corresponding to device n. This formulation aims to minimize a global loss function using data stored locally at every device, by training a model $\boldsymbol{x} \in \mathbb{R}^d$.

A. First-Order Methods

The idea behind solving this problem using first-order methods is to iteratively update the model in the opposite direction of the sum of the gradients received from all devices, using an appropriate learning rate, as

$$\boldsymbol{x}^{r+1} = \; \boldsymbol{x}^r - \alpha\nabla f(\boldsymbol{x}^r) = \boldsymbol{x}^r - \tfrac{\alpha}{N}\textstyle\sum_{n=1}^{N}\nabla f_n(\boldsymbol{x}^r), \tag{2}$$

where $\nabla f(\boldsymbol{x}^r)$ is the gradient of $f(\boldsymbol{x}^r)$ and α is the learning rate. Another way to implement this solution is to apply model aggregation at the PS instead of gradient aggregation expressed in (2). For this, every device computes one gradient step to update a local version of the model $\boldsymbol{x}_n^{r+1}$ using the local gradient, as follows,

$$\boldsymbol{x}_n^{r+1} = \; \boldsymbol{x}_n^r - \alpha\nabla f(\boldsymbol{x}_n^r). \tag{3}$$

At the PS side, the model $\boldsymbol{x}^{r+1}$ is updated by aggregating all the local models such that,

$$\boldsymbol{x}^{r+1} = \; \tfrac{1}{N}\textstyle\sum_{n=1}^{N}\boldsymbol{x}_n^{r+1} = \boldsymbol{x}^r - \tfrac{\alpha}{N}\textstyle\sum_{n=1}^{N}\nabla f_n(\boldsymbol{x}^r). \tag{4}$$

From the implementation in (4), instead of performing one local update step, every device can apply multiple local gradient updates before sending the model to the PS. This approach is called *FedAvg* [5].

B. Second-Order Methods

Contrary to first-order methods where only the gradient-type updates are computed, second-order methods benefit from the second-order information such as local curvature to speed up convergence by additionally utilizing the Hessian matrix to update the model. Applying Newton's method, at iteration $(r+1)$, the solution to problem (1) can be expressed as follows

$$\boldsymbol{x}^{r+1} = \boldsymbol{x}^r - \left(\frac{1}{N}\sum_{n=1}^{N}\nabla^2 f_n(\boldsymbol{x}^r)\right)^{-1}\left(\frac{1}{N}\sum_{n=1}^{N}\nabla f_n(\boldsymbol{x}^r)\right), \tag{5}$$

where $\nabla^2 f_n(\boldsymbol{x}^r) \in \mathbb{R}^{d\times d}$ is the Hessian of $f_n(\boldsymbol{x}^r)$. To solve this problem, *N* devices must exchange information with the PS through many iterations until convergence. During this process, we note the following: i) at every iteration *r*, every device *n* locally computes the gradient and the Hessian matrix, ii) the PS requires the transmission of both gradients and Hessian matrices from all devices to update the model $\boldsymbol{x}^{r+1}$, which incurs high communication cost, iii) the raw gradients and Hessians provide too much information about the local data and function, making them vulnerable to inverse attacks, giving rise to a privacy issue [24].

FedNL was proposed in [31] to provide a communication-efficient Hessian learning technique where, instead of sharing the exact Hessian to the PS, a compressed version of the Hessian is utilized. For example, if the rank-1 approximation of the Hessian is used, every device needs to compute the singular value decomposition (SVD) on the Hessian matrix

and then send the gradient along with two vectors for the compressed Hessian. Despite the reduction in communication costs, two observations are in order: (i) the compression (the number of shared vectors from the devices) is determined by the rank of the Hessian, (ii) the gradients and the compressed version of Hessian still reveal information about the function and training data, (iii) every device executes SVD locally at each iteration, which incurs high computational complexity (quadratic complexity). The authors in [31] also proposed an algorithm named *Newton-zero* to reduce the local computational complexity compared to *FedNL*. In this algorithm, the model $\boldsymbol{x}^{r+1}$ is updated at the PS as follows,

$$\boldsymbol{x}^{r+1} = \boldsymbol{x}^{r} - \left(\frac{1}{N}\sum_{n=1}^{N}\nabla^2 f_n(\boldsymbol{x}^0)\right)^{-1}\left(\frac{1}{N}\sum_{n=1}^{N}\nabla f_n(\boldsymbol{x}^r)\right). \tag{6}$$

We note that the Hessian matrix is computed and shared only at the first iteration ($r = 0$), which minimizes the local computational complexity. Nonetheless, sharing the initial Hessian matrix requires more communication resources compared to gradient sharing. In addition, privacy concerns still exist since the Hessian at the initial model is shared, as well as the gradient which is also shared at each iteration. Unlike *FedNL* and *Newton-zero* methods, we avoid sending the Hessian or its compressed version at any iteration. As we will explain next, we share only one vector that hides the true gradient and Hessian information.

C. Proposed Federated Second-Order Newton Approach

To enable communication-efficient FL and alleviate privacy concerns when transmitting both gradients and the Hessian matrix, we propose a framework that enables each device to transmit only a vector to the PS in every communication round. This leads to the same communication cost as the first-order distributed algorithms, while being privacy-preserving since neither the gradients nor the Hessian matrices are shared with the PS.

Let $\boldsymbol{H}_n^0 = \nabla^2 f_n(\boldsymbol{x}^0)$ be the Hessian matrix of device n at the initial iteration $r = 0$. Hence, the average of the Hessian matrices from all devices, referred to as system Hessian, is $\boldsymbol{H}^0 = \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{H}_n^0$. Moreover, let $\boldsymbol{g}_n^r = \nabla f_n(\boldsymbol{x}^r)$ be the gradient vector at device n, and $\boldsymbol{g}^r = \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{g}_n^r$ be the system gradient. Therefore, the updating step of (6) can be written as,

$$\boldsymbol{x}^{r+1} = \boldsymbol{x}^{r} - (\boldsymbol{H}^0)^{-1}\boldsymbol{g}^r. \tag{7}$$

The second term at the right hand side of (7) is calculated at every iteration r. Consequently, this *inverse Hessian-gradient product* term can be set as the solution to the following quadratic optimization problem,

$$\boldsymbol{w}^r = \underset{\boldsymbol{w}\in\mathbb{R}^d}{\arg\min}\ \frac{1}{2}\boldsymbol{w}^T\boldsymbol{H}^0\boldsymbol{w} - \boldsymbol{w}^T\boldsymbol{g}^r. \tag{8}$$

In other words, solving problem (8) results in $\boldsymbol{w}^r = (\boldsymbol{H}^0)^{-1}\boldsymbol{g}^r$. However, this problem cannot be solved in a distributed manner. To this end, we propose to reformulate (8) and cast it as a distributed optimization problem as follows,

$$\underset{\{\mathbf{w}_n,\boldsymbol{w}\}\in\mathbb{R}^d}{\arg\min}\ \sum_{n=1}^{N}\frac{1}{2}\mathbf{w}_n^T\boldsymbol{H}_n^0\mathbf{w}_n - \mathbf{w}_n^T\boldsymbol{g}_n^r \quad s.t.\ \ \mathbf{w}_n = \boldsymbol{w},\ \ \forall n, \tag{9}$$

where $\boldsymbol{w}$ is updated at the PS and $\boldsymbol{w}_n$ is the local version for device n. Hence, we formulate our proposed approach based on (7) and (9) as a bilevel optimization problem consisting of: i) *inner-level problem*: where the problem in (9) is solved by all devices to obtain $\boldsymbol{w}^r = (\boldsymbol{H}^0)^{-1}\boldsymbol{g}^r$, and ii) *outer-level loop*: we leverage the solution of the *inner-level problem* to update the global model $\boldsymbol{x}^{r+1}$ using (7).

However, solving (9) exactly at every *outer-level* (Newton) iteration may come at a huge communication cost. Hence, to account for communication overhead, we solve the *inner-level problem* by performing a few ADMM updates to approximate the *inverse-Hessian-gradient product*, which is used by the PS to update the global model. In the following, we describe the ADMM framework used to solve the *inner-level problem*. Since the objective function in problem (9) is separable across devices, we use the Augmented Lagrangian to solve it as a sequence of unconstrained subproblems. The Augmented Lagrangian of problem (9) is defined as follows,

$$\mathcal{L}_\rho(\boldsymbol{w},\{\mathbf{w}_n\}_{n=1}^N,\boldsymbol{\lambda}) = \sum_{n=1}^{N}\left[\frac{1}{2}\mathbf{w}_n^T\boldsymbol{H}_n^0\mathbf{w}_n - \mathbf{w}_n^T\boldsymbol{g}_n^r\right] + \sum_{n=1}^{N}\langle\boldsymbol{\lambda}_n,\mathbf{w}_n-\boldsymbol{w}\rangle + \frac{\rho}{2}\sum_{n=1}^{N}\|\mathbf{w}_n-\boldsymbol{w}\|^2, \tag{10}$$

where $\boldsymbol{\lambda}_n$ is the dual variable for device n that is updated at every step, $\rho > 0$ is the constant that controls the mismatch between $\boldsymbol{w}$ and the local version $\boldsymbol{w}_n$, and $\langle .,. \rangle$ denotes the inner product operation.

At *outer-level* iteration r and *inner-level* ADMM step $(k+1)$, every device n updates its local version by minimizing $\mathcal{L}_\rho(\boldsymbol{w}^{r,k},\boldsymbol{w}_n,\boldsymbol{\lambda}^{r,k})$. Thus, the local version $\boldsymbol{w}_n^{r,k+1}$ should satisfy the following,

$$\mathbf{w}_n^{r,k+1} = \underset{\mathbf{w}_n}{\arg\min}\ \mathcal{L}_\rho\Big(\boldsymbol{w}^{r,k},\mathbf{w}_n,\boldsymbol{\lambda}^{r,k}\Big). \tag{11}$$

Setting the derivative equal to zero,

$$\frac{d\mathcal{L}_\rho\Big(\boldsymbol{w}^{r,k},\mathbf{w}_n^{r,k+1},\boldsymbol{\lambda}^{r,k}\Big)}{d\mathbf{w}_n^{r,k+1}} = \mathbf{0},\quad \forall\ n\in[N], \tag{12}$$

yields the following equation,

$$\boldsymbol{H}_n^0\mathbf{w}_n^{r,k+1} - \boldsymbol{g}_n^r + \boldsymbol{\lambda}_n^{r,k} + \rho\Big(\mathbf{w}_n^{r,k+1} - \boldsymbol{w}^{r,k}\Big) = \mathbf{0}. \tag{13}$$

Hence, it results in the closed-form expression,

$$\mathbf{w}_n^{r,k+1} = \Big(\boldsymbol{H}_n^0 + \rho\boldsymbol{I}\Big)^{-1}\Big(\boldsymbol{g}_n^r - \boldsymbol{\lambda}_n^{r,k} + \rho\boldsymbol{w}^{r,k}\Big). \tag{14}$$

After that, every device n sends $\boldsymbol{w}_n^{r,k+1}$ to the PS, where the latter updates $\boldsymbol{w}^{r,k+1}$ by solving the following optimization problem

$$\boldsymbol{w}^{r,k+1} = \underset{\boldsymbol{w}}{\arg\min}\ \mathcal{L}_\rho\Big(\boldsymbol{w},\{\mathbf{w}_n^{r,k+1}\}_{n=1}^N,\boldsymbol{\lambda}^{r,k}\Big). \tag{15}$$

Setting the derivative to zero gives the following expression,

$$\sum_{n=1}^{N} \frac{d\langle \boldsymbol{\lambda}_n^{r,k}, \mathbf{w}_n^{r,k+1} - \boldsymbol{w}^{r,k+1}\rangle}{d\boldsymbol{w}^{r,k+1}} + \sum_{n=1}^{N} \frac{\rho}{2} \frac{d\left\|\mathbf{w}_n^{r,k+1} - \boldsymbol{w}^{r,k+1}\right\|^2}{d\boldsymbol{w}^{r,k+1}} = \mathbf{0}, \tag{16}$$

and the update of $\boldsymbol{w}^{r,k+1}$ is given by

$$\boldsymbol{w}^{r,k+1} = \frac{1}{N} \sum_{n=1}^{N} \left[\mathbf{w}_n^{r,k+1} + \frac{\boldsymbol{\lambda}_n^{r,k}}{\rho} \right]. \tag{17}$$

Next, the PS broadcasts $\boldsymbol{w}^{r,k+1}$ to the devices and then the dual variable $\boldsymbol{\lambda}_n^{r,k+1}$ is updated locally such that,

$$\boldsymbol{\lambda}_n^{r,k+1} = \boldsymbol{\lambda}_n^{r,k} + \rho\left(\mathbf{w}_n^{r,k+1} - \boldsymbol{w}^{r,k+1}\right). \tag{18}$$

Using (17) and taking the sum of the dual variable updates in (18) across devices gives the following expression,

$$\sum_{n=1}^{N} \boldsymbol{\lambda}_n^{r,k+1} = \sum_{n=1}^{N} \boldsymbol{\lambda}_n^{r,k} + \rho\left(\sum_{n=1}^{N} \mathbf{w}_n^{r,k+1} - \sum_{n=1}^{N} \left[\mathbf{w}_n^{r,k+1} + \frac{\boldsymbol{\lambda}_n^{r,k}}{\rho} \right] \right). \tag{19}$$

Consequently, we get

$$\sum_{n=1}^{N} \boldsymbol{\lambda}_n^{r,k+1} = \mathbf{0}. \tag{20}$$

Hence, the update of $\boldsymbol{w}^{r,k+1}$ is simply the average of the local versions received from the devices,

$$\boldsymbol{w}^{r,k+1} = \frac{1}{N} \sum_{n=1}^{N} \mathbf{w}_n^{r,k+1}. \tag{21}$$

After running K inner ADMM steps, the model $\boldsymbol{x}^{r+1}$ is updated following (7) such that,

$$\boldsymbol{x}^{r+1} = \boldsymbol{x}^{r} - \boldsymbol{w}^{r,K}. \tag{22}$$

It is worth mentioning that the updates at $k = 0$ are set as: $\boldsymbol{w}^{r,0} = \boldsymbol{w}^{r-1,K}$, $\boldsymbol{w}_n^{r,0} = \boldsymbol{w}_n^{r-1,K}$, and $\boldsymbol{\lambda}_n^{r,0} = \boldsymbol{\lambda}_n^{r-1,K}$.

III. Analog Over-the-Air Model Aggregation for Second-Order Type Newton Method

From the update of $\boldsymbol{w}^{r,k+1}$ at the PS in (21), we notice that the aggregation of the local versions motivates the implementation of *analog over-the-air aggregation* strategies since we can take advantage of the superposition property at the PS. This is done by sending the i^{th} element of signals from all devices using the same subcarrier. This significantly reduces the communication overhead, speeds up convergence, and provides scalability. In contrast, the digital scheme necessitates orthogonal bandwidth allocation resulting in a communication cost that is directly proportional to the number of devices.

To solve problem (9) using *analog over-the-air aggregation*, we adopt two existing algorithms that were used for first-order methods [34], [35] and extend them further using second-order formulation. In what follows, we define the two proposed algorithms:

- *NAAM-v0:* To cancel the effect of the channel, every device *n* divides its i^{th} element by its corresponding channel coefficient $h_{n,i}$ before transmission. The device is able to transmit the i^{th} element only when $|h_{n,i}| > \beta$, where β is a predefined threshold,
- *NAAM-v1:* Instead of performing channel inversion that can hinder the system performance, we benefit from the non-channel inversion algorithm [34] and integrate the channel effect into our problem formulation.

A. NAAM-v0

With the assumption of flat fading channel per subcarrier, the channel output received by the PS at every time slot *t* over the i^{th} subcarrier is expressed as,

$$g_i(t) = \sum_{n=1}^{N} h_{n,i}(t) \cdot s_{n,i}(t) + z_i(t), \tag{23}$$

where $h_{n,i}(t)$ is the flat fading channel at the i^{th} subcarrier between device *n* and the PS at time *t*, $s_{n,i}(t)$ is the i^{th} signal from device *n*, and $z_i(t)$ is the additive white Gaussian (AWGN) noise at the PS at time *t* following $\mathcal{CN}(0,1)$. In order to update $w_i^{r,k+1}$ at the PS, every device *n*, after updating $w_{n,i}^{r,k+1}$ using (14), transmits it to the PS for a duration of *T* seconds. Hence, for every element *i*, the PS needs to apply the update shown below,

$$w_i^{r,k+1} = \mathbf{Re}\left\{ \frac{1}{N} \sum_{n=1}^{N} w_{n,i}^{r,k+1} + \hat{z}_i^{r,k+1} \right\}, \tag{24}$$

where **Re**(.) is the real part of a complex value. Since the local versions are real-valued, we get

$$w_i^{r,k+1} = \frac{1}{N} \sum_{n=1}^{N} w_{n,i}^{r,k+1} + \mathbf{Re}\left\{ \hat{z}_i^{r,k+1} \right\}. \tag{25}$$

To guarantee this, we need to perform two operations at the devices' side: (i) channel inversion to cancel the channel effect, and (ii) power control such that every device's transmit power budget is satisfied and equal power for all the devices is attained at the PS side.

To mitigate the effect of the channel, every device *n* divides its i^{th} element by the channel coefficient $h_{n,i}$. However, for deep fading channel, the channel inverse operation can lead to power budget violation, thus we set the transmission decision as

$$s_{n,i}(t) = \begin{cases} c(t) \frac{w_{n,i}^{r,k+1}(t)}{h_{n,i}(t)} & \text{if } |h_{n,i}(t)| > \beta \\ 0 & \text{otherwise} \end{cases} \tag{26}$$

where *c*(*t*) is the scaling factor to be calculated at the PS, as detailed next. We denote the set of devices that can transmit over the i^{th} subcarrier by

$$\bar{N}_i(t) = \left\{ n \in [N] : |h_{n,i}(t)| > \beta \right\} \quad \forall i \in [d], \tag{27}$$

and for every device *n*, the set of elements that can be transmitted which satisfies the condition in (26) is like this,

$$\bar{d}_n(t) = \left\{ i \in [d] : |h_{n,i}(t)| > \beta \right\} \quad \forall n \in [N]. \tag{28}$$

Therefore, every device n needs to calculate its scaling factor $\alpha_n(t)$ such that,

$$\frac{c_n(t)^2}{\bar{d}_n}\sum_{i=1}^{\bar{d}_n}\left|\frac{w_{n,i}^{r,k+1}(t)}{h_{n,i}(t)}\right|^2 \leq P_n, \tag{29}$$

where P_n is the maximum power. After that, every device n sends the scalar $c_n(t)$ in an error free mode to the PS, and the latter determines $c(t)$ as:

$$c(t) = \min\Big\{c_1(t), \ldots, c_{\bar{N}_i}(t)\Big\}, \tag{30}$$

Next, the PS broadcasts the scaling factor $c(t)$ through a control channel to the devices. Subsequently, every device transmits $c(t)\frac{w_{n,i}^{r,k+1}(t)}{h_{n,i}(t)}$ over the i^{th} subcarrier for a duration of T seconds. The PS receives $\sum_{n=1}^{\bar{N}_i(t)} c(t) w_{n,i}^{r,k+1}(t) + z_i^{r,k+1}(t)$ for every $t \in [T]$. Finally, after dividing by $c(t)$, matched filtering is performed such that the received signals are integrated during T seconds, divided by T, and sampled at $t = T$. This produces $\sum_{n=1}^{\bar{N}_i} w_{n,i}^{r,k+1} + \hat{z}_i^{r,k+1}$ with reduced noise $\hat{z}_i^{r,k+1} \sim \mathcal{CN}(0, N_0/T)$, where N_0 is the AWGN power spectral density. Consequently, the PS constructs the update in (24). In the downlink, every device n receives $w_i^{r,k+1} + \hat{z}_{n,i}^{r,k+1}$, and the dual variables are updated as follows,

$$\lambda_{n,i}^{r,k+1} = \lambda_{n,i}^{r,k} + \rho\Big(w_{n,i}^{r,k+1} - w_i^{r,k+1}\Big) - \rho\Big\{\hat{z}_{n,i}^{r,k+1}\Big\}. \tag{31}$$

B. NAAM-v1

The *NAAM-v1* implementation natively incorporates the channel in the formulation of the optimization problem to avoid channel inversion and avoid choosing β, which is a hyper-parameter that can highly affect the system performance. Concretely, we reformulate the problem in (9) by introducing the channel in the constraint as follows

$$\begin{aligned}\underset{\{\mathbf{w}_n, \boldsymbol{w}\}\in\mathbb{R}^d}{\arg\min} \; & \sum_{n=1}^{N}\frac{1}{2}\mathbf{w}_n^T \boldsymbol{H}_n^0 \mathbf{w}_n - \mathbf{w}_n^T \boldsymbol{g}_n^r \\ s.t. \;\; & h_{n,i} w_{n,i} = h_{n,i} w_i, \;\; \forall n, i.\end{aligned} \tag{32}$$

Note that the constraint in (32) is equivalent to the one in (9) since multiplying both sides with the channel coefficient does not change it. This allows the integration of the channel effect into the formulation. For now, we assume that the channel is static and noise-free. Later in this section, we relax these assumptions. The augmented Lagrangian for problem (32) is formed as follows

$$\begin{aligned}\mathcal{L}_\rho\big(\boldsymbol{w}, \{\mathbf{w}_n\}_{n=1}^N, \boldsymbol{\lambda}\big) &= \sum_{n=1}^{N}\frac{1}{2}\mathbf{w}_n^T \boldsymbol{H}_n^0 \mathbf{w}_n - \mathbf{w}_n^T \boldsymbol{g}_n^r \\ &+ \sum_{n=1}^{N}\sum_{i=1}^{d}\lambda_{n,i}^* h_{n,i}(w_{n,i} - w_i) + \frac{\rho}{2}\sum_{n=1}^{N}\sum_{i=1}^{d}|h_{n,i}|^2(w_{n,i} - w_i)^2, \\ &= \sum_{n=1}^{N}\sum_{i=1}^{d}\left[w_{n,i}\Big(\frac{1}{2}\sum_{j=1}^{d}H_{n,i,j}^0 w_{n,j} - g_{n,i}^r\Big)\right] \\ &+ \sum_{n=1}^{N}\sum_{i=1}^{d}\lambda_{n,i}^* h_{n,i}(w_{n,i} - w_i) + \frac{\rho}{2}\sum_{n=1}^{N}\sum_{i=1}^{d}|h_{n,i}|^2(w_{n,i} - w_i)^2,\end{aligned}$$

where $\lambda_{n,i}^*$ is the complex conjugate of the dual variable $\lambda_{n,i}$ and $H_{n,i,j}^0$ is the element of the initial Hessian at iteration ($r = 0$) over the i^{th} row and the j^{th} column at device n. At step $(k+1)$, every device n updates its local version $w_{n,l}^{r,k+1}$ by minimizing $\mathcal{L}_\rho(\boldsymbol{w}^{r,k}, \boldsymbol{w}_n, \boldsymbol{\lambda}^{r,k})$. Thus, $w_{n,l}^{r,k+1}$ should satisfy

$$\frac{d\mathcal{L}_\rho\Big(\boldsymbol{w}^{r,k}, \mathbf{w}_n^{r,k+1}, \boldsymbol{\lambda}^{r,k}\Big)}{dw_{n,l}^{r,k+1}} = 0, \quad \forall\, n \in [N], l \in [d]. \tag{34}$$

Next, we introduce the vector $\boldsymbol{e} \in \mathbb{R}^d$ whose l^{th} element is given by

$$\begin{aligned}\boldsymbol{e}_l &= \frac{d\mathcal{L}_\rho\Big(\boldsymbol{w}^{r,k}, \mathbf{w}_n^{r,k+1}, \boldsymbol{\lambda}^{r,k}\Big)}{dw_{n,l}^{r,k+1}} \\ &= \frac{1}{2}\sum_{i=1}^{d}\frac{dw_{n,i}^{r,k+1}}{dw_{n,l}^{r,k+1}}\sum_{j=1}^{d}H_{n,i,j}^0 w_{n,j}^{r,k+1} \\ &+ \frac{1}{2}\sum_{i=1}^{d}w_{n,i}^{r,k+1}\sum_{j=1}^{d}H_{n,i,j}^0\frac{dw_{n,j}^{r,k+1}}{dw_{n,l}^{r,k+1}} - \sum_{i=1}^{d}\frac{dw_{n,i}^{r,k+1}}{dw_{n,l}^{r,k+1}}g_{n,i}^r \\ &+ \sum_{i=1}^{d}\Big(\lambda_{n,i}^{r,k}\Big)^* h_{n,i}\frac{d\Big(w_{n,i}^{r,k+1} - w_i^{r,k}\Big)}{dw_{n,l}^{r,k+1}} \\ &+ \frac{\rho}{2}\sum_{i=1}^{d}|h_{n,i}|^2\frac{d\Big(w_{n,i}^{r,k+1} - w_i^{r,k}\Big)^2}{dw_{n,l}^{r,k+1}} \\ &= \frac{1}{2}\sum_{j=1}^{d}H_{n,l,j}^0 w_{n,j}^{r,k+1} + \frac{1}{2}\sum_{i=1}^{d}w_{n,i}^{r,k+1}H_{n,i,l}^0 - g_{n,l}^r \\ &\quad + \Big(\lambda_{n,l}^{r,k}\Big)^* h_{n,l} \;\; + \rho|h_{n,l}|^2\Big(w_{n,l}^{r,k+1} - w_l^{r,k}\Big).\end{aligned} \tag{35}$$

Therefore, we can write $\boldsymbol{e}$ in a compact form as shown below,

$$\begin{aligned}\boldsymbol{e} = \frac{1}{2}(&\boldsymbol{H}_n^0\mathbf{w}_n^{r,k+1} + \Big(\boldsymbol{H}_n^0\Big)^T\mathbf{w}_n^{r,k+1} - \boldsymbol{g}_n^r \\ &+ \Big(\boldsymbol{\lambda}_n^{r,k}\Big)^* \circ \boldsymbol{h_n} + \rho((\boldsymbol{h_n})^* \circ \boldsymbol{h_n}) \circ \Big(\mathbf{w}_n^{r,k+1} - \boldsymbol{w}^{r,k}\Big),\end{aligned} \tag{36}$$

where $\circ$ is the Hadamard product operation and $(.)^T$ is the transpose operation. We set $\bar{\boldsymbol{h}}_n = (\boldsymbol{h_n})^* \circ \boldsymbol{h_n}$. Equivalently, we can write equation (36) as follows

$$\begin{aligned}\boldsymbol{e} = \boldsymbol{H}_n^0\mathbf{w}_n^{r,k+1} - \boldsymbol{g}_n^r + \Big(\boldsymbol{\lambda}_n^{r,k}\Big)^* \circ \boldsymbol{h_n} \\ + \;\; \rho\bar{\boldsymbol{h}_n} \circ \Big(\mathbf{w}_n^{r,k+1} - \boldsymbol{w}^{r,k}\Big).\end{aligned} \tag{37}$$

Theorem 1: Let $\boldsymbol{a}$ and $\boldsymbol{b}$ be real vectors of size d. Then $\boldsymbol{a} \circ \boldsymbol{b} = \mathrm{diag}(\boldsymbol{a})\boldsymbol{b}$, where $\mathrm{diag}(\boldsymbol{a})$ is the $d \times d$ diagonal matrix whose entries are the d elements of the vector $\boldsymbol{a}$.

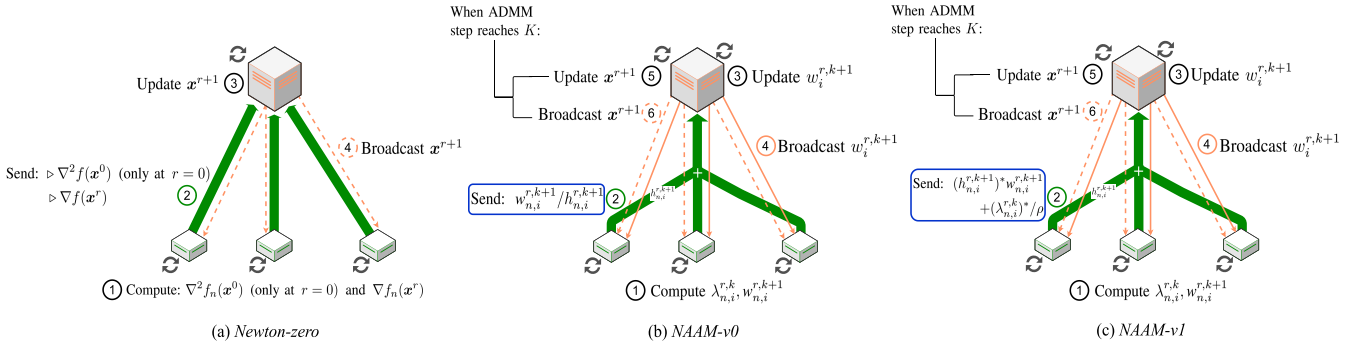
Proof: Let $\boldsymbol{a} \in \mathbb{R}^d$ and $\boldsymbol{b} \in \mathbb{R}^d$. Then $[\mathrm{diag}(\boldsymbol{a})\boldsymbol{b}]_i = [\boldsymbol{a}]_i[\boldsymbol{b}]_i = [\boldsymbol{a} \circ \boldsymbol{b}]_i$ and therefore, $\boldsymbol{a} \circ \boldsymbol{b} = \mathrm{diag}(\boldsymbol{a})\boldsymbol{b}$. ■

Hence, setting the derivative equal to zero, we get,

$$\begin{aligned}&\boldsymbol{H}_n^0\mathbf{w}_n^{r,k+1} - \boldsymbol{g}_n^r + \big(\boldsymbol{\lambda}_n^{r,k}\big)^* \circ \boldsymbol{h_n} + \rho\,\mathrm{diag}(\bar{\boldsymbol{h}}_n)\big(\mathbf{w}_n^{r,k+1} - \boldsymbol{w}^{r,k}\big) \\ &= \mathbf{0}.\end{aligned} \tag{38}$$

Equivalently, by arranging the terms, we can write,

$$\begin{aligned}&\big[\boldsymbol{H}_n^0 + \rho\,\mathrm{diag}\big(\bar{\boldsymbol{h}}_n\big)\big]\mathbf{w}_n^{r,k+1} - \boldsymbol{g}_n^r + \big(\boldsymbol{\lambda}_n^{r,k}\big)^* \circ \boldsymbol{h_n} - \rho\,\mathrm{diag}\big(\bar{\boldsymbol{h}}_n\big)\boldsymbol{w}^{r,k} \\ &= \mathbf{0}.\end{aligned} \tag{39}$$



(a) *Newton-zero* (b) *NAAM-v0* (c) *NAAM-v1*

Fig. 1. Schematic illustrations of the algorithms: (a) *Newton-zero*, (b) *NAAM-v0*, and (c) *NAAM-v1*.

As a result, $\boldsymbol{w}_n^{r,k+1}$ is updated as follows

$$\mathbf{w}_n^{r,k+1} = \Big[\boldsymbol{H}_n^0 + \rho\,\mathrm{diag}\big(\bar{\boldsymbol{h}}_n\big)\Big]^{-1}\Big[\boldsymbol{g}_n^r - \Big(\boldsymbol{\lambda}_n^{r,k}\Big)^* \circ \boldsymbol{h_n} + \rho\,\mathrm{diag}\big(\bar{\boldsymbol{h}}_n\big)\boldsymbol{w}^{r,k}\Big], \tag{40}$$

and $w_{n,i}^{r,k+1}$ corresponds to the i^{th} entry of the update in (40). After updating the local version $w_{n,i}^{r,k+1}$ at every device n, the PS collects all the local versions from the devices and updates the i^{th} element $w_i^{r,k+1}$ by minimizing $\mathcal{L}_\rho(\boldsymbol{w}, \{\boldsymbol{w}_n^{r,k+1}\}_{n=1}^N, \boldsymbol{\lambda}^{r,k})$. Thus, $w_i^{r,k+1}$ should satisfy

$$\frac{d\mathcal{L}_\rho\Big(\boldsymbol{w}^{r,k+1}, \Big\{\mathbf{w}_n^{r,k+1}\Big\}_{n=1}^N, \boldsymbol{\lambda}^{r,k}\Big)}{dw_i^{r,k+1}} = 0, \quad \forall\ i \in [d], \tag{41}$$

which yields the following update,

$$w_i^{r,k+1} = \frac{1}{\sum_{n=1}^N |h_{n,i}|^2}\sum_{n=1}^N \Big(|h_{n,i}|^2 w_{n,i}^{r,k+1} + \Big(\lambda_{n,i}^{r,k}\Big)^* h_{n,i}/\rho\Big). \tag{42}$$

In order for the PS to apply equation (42), every device n needs to transmit $h_{n,i}^* w_{n,i}^{r,k+1} + (\lambda_{n,i}^{r,k})^*/\rho$ for a duration of T seconds. After that, the PS broadcasts $w_i^{r,k+1}$ to the devices. Device n updates its dual variable as follows,

$$\lambda_{n,i}^{r,k+1} = \lambda_{n,i}^{r,k} + \rho h_{n,i}\Big(w_{n,i}^{r,k+1} - w_i^{r,k+1}\Big). \tag{43}$$

A schematic representation of *Newton-zero*, *NAAM-v0*, and *NAAM-v1* is depicted in Fig. 1.

1) Time-Varying and Noisy Channel: The implementation of the algorithm needs to account for channel dynamics and noise. First, we consider the case when $h_{n,i}^{r,k} = h_{n,i}^{r,k+1}$ for all $n \in [N]$ and $i \in [d]$, which indicates that the channel at iterations k and $k+1$ remains invariant between the PS and the devices. Second, we introduce an additional step to account for the case when the channel changes, i.e., when $h_{n,i}^{r,k+1} \neq h_{n,i}^{r,k}$.

To update $w_i^{r,k+1}$, every device n transmits $(h_{n,i}^{r,k+1})^* w_{n,i}^{r,k+1} + (\lambda_{n,i}^{r,k})^*/\rho$ over T seconds. At the PS side, the received signal from all the devices is $\sum_{n=1}^N (|h_{n,i}^{r,k+1}|^2 w_{n,i}^{r,k+1} + (\lambda_{n,i}^{r,k})^* h_{n,i}^{r,k+1}/\rho) + z_i^{r,k+1}(t)$ for every second $t \in [0, T]$, where $z_i^{r,k+1}(t) \sim \mathcal{CN}(0, N_0)$ is the noise term at the PS side. As illustrated in Fig. 2, matched filtering is then applied at the PS by integrating the signal during T seconds and sampling at $t = T$. This results in $\sum_{n=1}^N (|h_{n,i}^{r,k+1}|^2 w_{n,i}^{r,k+1} + (\lambda_{n,i}^{r,k})^* h_{n,i}^{r,k+1}/\rho) + \hat{z}_i^{r,k+1}$ with reduced noise $\hat{z}_i^{r,k+1}(t) \sim \mathcal{CN}(0, N_0/T)$. Hence, $w_i^{r,k+1}$ is updated as follows,

$$w_i^{r,k+1} = \frac{1}{\sum_{n=1}^N |h_{n,i}^{r,k+1}|^2}\left[\sum_{n=1}^N \left(\left|h_{n,i}^{r,k+1}\right|^2 w_{n,i}^{r,k+1} + \left(\lambda_{n,i}^{r,k}\right)^* h_{n,i}^{r,k+1}/\rho\right) + \mathbf{Re}\Big\{\hat{z}_i^{r,k+1}\Big\}\right], \tag{44}$$

where taking the real part of the reduced noise term follows from the fact that the update $w_i^{r,k+1}$ is real-valued. In the downlink, every device receives $h_{n,i}^{k+1} w_i^{r,k+1} + \hat{z}_{n,i}^{r,k+1}$ from the PS and then locally multiplies it by $(h_{n,i}^{k+1})^*$ to fit it in the update of $w_{n,i}^{r,k+1}$. We write the update in vector form as seen below,

$$\mathbf{w}_n^{r,k+1} = \Big[\boldsymbol{H}_n^0 + \rho\,\mathrm{diag}(\bar{\boldsymbol{h}}_n^{r,k+1})\Big]^{-1}\Big[\boldsymbol{g}_n^r - \Big(\boldsymbol{\lambda}_n^{r,k}\Big)^* \circ \boldsymbol{h}_n^{r,k+1} + \rho\,\mathrm{diag}\Big(\bar{\boldsymbol{h}}_n^{r,k+1}\Big)\boldsymbol{w}^{r,k} + \rho\,(\boldsymbol{h}^{r,k+1})^* \circ \hat{\boldsymbol{z}}_n^{r,k}\Big]. \tag{45}$$

Finally, the dual variable is updated as follows,

$$\lambda_{n,i}^{r,k+1} = \lambda_{n,i}^{r,k} + \rho h_{n,i}^{r,k+1}\Big(w_{n,i}^{r,k+1} - w_i^{r,k+1}\Big) - \rho\hat{z}_{n,i}^{r,k+1}. \tag{46}$$

In the case when $h_{n,i}^{r,k+1} \neq h_{n,i}^{r,k}$, similarly to [34], we first use the previous value of the local version, i.e., $w_{n,i}^{r,k+1} = w_{n,i}^{r,k}$ and then use it to solve the equation (45) with respect of $(\lambda_{n,i}^{r,k})^*$. Thereby, we guarantee that the primal and dual updates cope with changes in the channel. The pseudocode is detailed in Algorithm 1.

2) Privacy Analysis: To update the model $\boldsymbol{x}^{r+1}$, devices need to approximate the *inverse Hessian-gradient product* term found on the right hand side of (7). For this, every device runs for K ADMM steps during which a vector is updated and shared between the devices and the PS. We recall the

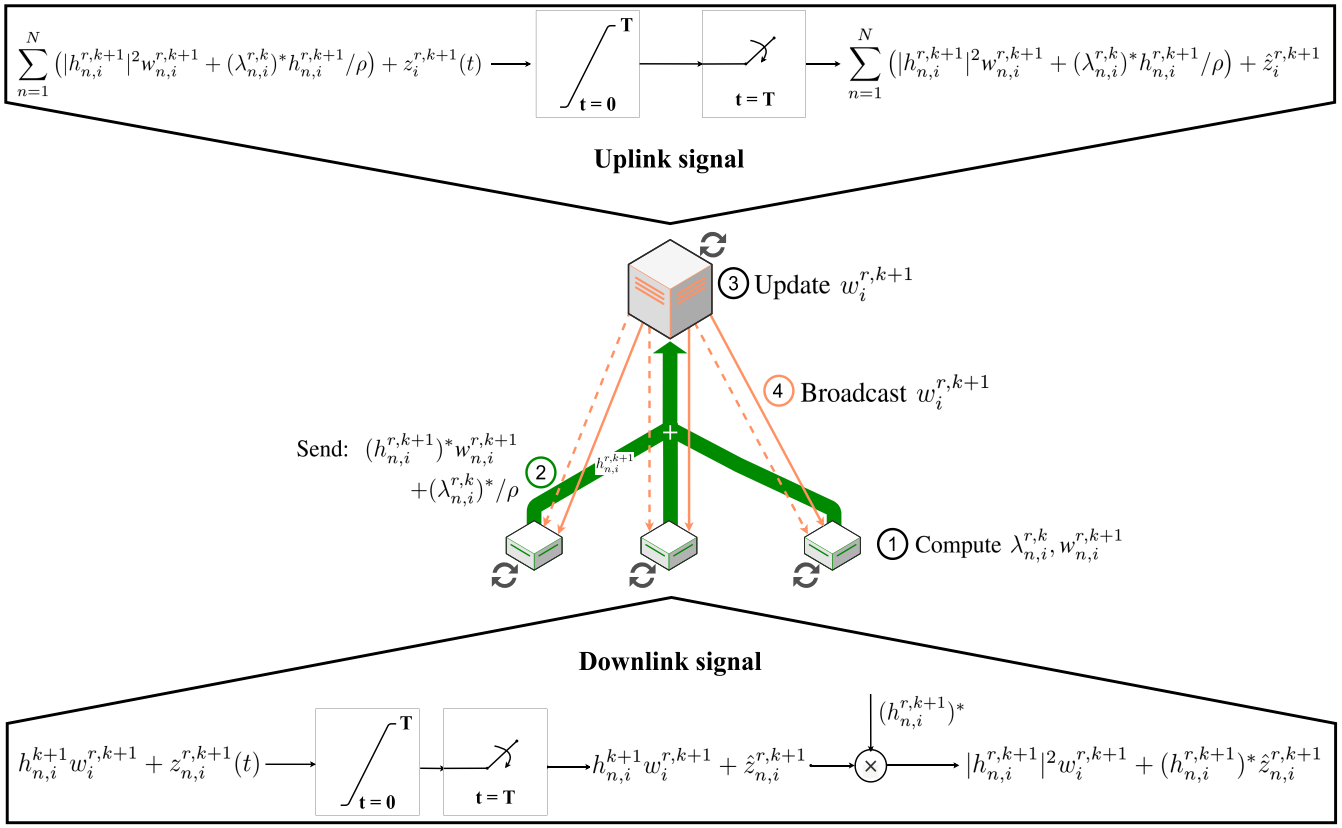


Fig. 2. Matched filtering illustration for *NAAM-v1* under noisy channel.

expression of $\boldsymbol{w}_n^{r,k+1}$ as follows,

$$\mathbf{w}_n^{r,k+1} = \Big[\boldsymbol{H}_n^0 + \rho\,\mathrm{diag}\Big(\bar{\boldsymbol{h}}_n^{r,k+1}\Big)\Big]^{-1}\Big[\boldsymbol{g}_n^r - \Big(\boldsymbol{\lambda}_n^{r,k}\Big)^* \circ \boldsymbol{h}_n^{r,k+1} + \rho\,\mathrm{diag}\Big(\bar{\boldsymbol{h}}_n^{r,k+1}\Big)\boldsymbol{w}^{r,k} + \rho\Big(\boldsymbol{h}_n^{r,k+1}\Big)^* \circ \hat{\boldsymbol{z}}_n^{r,k}\Big]. \tag{47}$$

As mentioned in Section III-B1, updating $\boldsymbol{w}^{r,k+1}$ means that the PS receives $\bar{\boldsymbol{h}}_n^{r,k+1} \circ \boldsymbol{w}_n^{r,k+1} + \frac{1}{\rho}(\boldsymbol{\lambda}_n^{r,k})^* \boldsymbol{h}_n^{r,k+1} + \mathbf{Re}\{\hat{\boldsymbol{z}}_n^{r,k+1}\}$ at every ADMM step $k+1$. Following the privacy analysis from [34], our algorithm *NAAM-v1* provides extra privacy protection, since the channel perturbations hide the update trajectory of $\boldsymbol{w}_n^{r,k+1}$, which represents neither the gradient nor the Hessian matrix.

IV. Numerical Results

In this section, we numerically evaluate the performance of our proposed approach and compare it to several baselines. We provide details about the considered learning problem, simulation settings, and obtained results.

Setting: We consider the regularized logistic regression problem defined as

$$\min_{\boldsymbol{x}\in\mathbb{R}^d} \left\{ f(\boldsymbol{x}) := \frac{1}{N}\sum_{n=1}^{N} f_n(\boldsymbol{x}) + \frac{\mu}{2}||\boldsymbol{x}||^2 \right\}, \tag{48}$$

where

$$f_n(\boldsymbol{x}) = \frac{1}{M}\sum_{m=1}^{M} \log\Big(1+\exp\Big(-b_{n,m}\,\boldsymbol{a}_{n,m}^T\boldsymbol{x}\Big)\Big), \tag{49}$$

TABLE II
Datasets Details

Dataset	N	M	$(N \times M)$	d
w7a	50	493	24650	300
w8a	142	350	49700	300
a9a	80	407	32560	123

μ is the regularization parameter. For this task, we use real datasets chosen from LibSVM where we consider different numbers of devices [36]. With that in mind, we study the performance of the proposed algorithm under different network sizes. We use $\{\boldsymbol{a}_{n,m}, b_{n,m}\}_{m\in[M]}$ to denote the data samples corresponding to the n^{th} device ($n \in \{1,\ldots,N\}$) where M is the number of samples allocated for every device. The details about datasets partitioning are listed in Table II.

Network Parameters and Communication Environment: Throughout the simulation, we set the transmitted power budget of each device to $P_n = 1\,\mathrm{mW}, \forall\, n \in \{1,\ldots,N\}$. By default, we consider a fixed number of available subcarriers, $N_s = 64$ with SNR $= 20\,dB$, where every subcarrier provides $W = 15\,\mathrm{KHz}$ of bandwidth for $\tau = 1\,\mathrm{ms}$ time duration according to [37]. We generate time-varying and fading channel coefficients between the agents and the PS following Rayleigh distribution with zero mean and unit variance, i.e., $h \sim \mathcal{CN}(0,1)$. We assume that every channel realization is coherent during 10 communication rounds.

Algorithm 1 *NAAM-v1*

Input: $N, f_n(\cdot), \rho, R, K, d, \quad \forall n$
Output: $\boldsymbol{x}$
Initialization: $\boldsymbol{x}^0, \boldsymbol{w}^{0,0}, \boldsymbol{w}_n^{0,0}, \boldsymbol{\lambda}_n^{0,0}, \quad \forall n.$
while $r \leq R$ **do**
  set: $\boldsymbol{w}^{r,0} = \boldsymbol{w}^{r-1,K}, \boldsymbol{w}_n^{r,0} = \boldsymbol{w}_n^{r-1,K}, \boldsymbol{\lambda}_n^{r,0} = \boldsymbol{\lambda}_n^{r-1,K}, \quad \forall n.$
  while $k \leq K$ **do**
    All devices in parallel:
    for $i = 1, \ldots, d$ **do**
      if $h_{n,i}^{r,k+1} = h_{n,i}^{r,k}$ **then**
        Find $w_{n,i}^{r,k+1}$ from (45)
      else
        $w_{n,i}^{r,k+1} \leftarrow w_{n,i}^{r,k}$
        Find $(\lambda_{n,i}^{r,k})^*$ from (45)
      end if
    end for
    Send $(h_{n,i}^{r,k+1})^* w_{n,i}^{r,k+1} + (\lambda_{n,i}^{r,k})^*/\rho, \forall i = 1, \cdots, d$ to the PS.
    PS:
    Find $w_i^{r,k+1}$ from (44)
    Broadcast $w_i^{r,k+1}, \forall i = 1, \ldots, d$ to all devices.
    All devices in parallel:
    Update $\lambda_{n,i}^{r,k+1}$ locally via (46), $\forall i = 1, \ldots, d$.
    $k \leftarrow k + 1$
  end while
  $\boldsymbol{x}^{r+1} \leftarrow \boldsymbol{x}^r - \boldsymbol{w}^{r,K}$
end while

In our simulations, we apply the analog communication scheme only on the uplink part to highlight the communication bottleneck in different settings, whereas digital scheme is utilized in the downlink. Therefore, the update of $\boldsymbol{w}^{r,k+1}$ for *NAAM-v1* under noisy channel is done by applying the analog scheme as per (44). For the downlink, the PS broadcasts $\boldsymbol{w}^{r,k+1}$ to the agents through separate channels. To update the local version $w_{n,i}^{r,k+1}$ and the dual variable $\lambda_{n,i}^{r,k+1}$, every device locally multiplies the received element by $|h_{n,i}^{r,k+1}|^2$ and $h_{n,i}^{r,k+1}$, respectively. The updates are shown as follows

$$\mathbf{w}_n^{r,k+1} = \left[\boldsymbol{H}_n^0 + \rho \operatorname{diag}\left(\bar{\boldsymbol{h}}_n^{r,k+1}\right)\right]^{-1} \left[\boldsymbol{y}_n^r - \left(\boldsymbol{\lambda}_n^{r,k}\right)^* \circ \boldsymbol{h}_n^{r,k+1} + \rho \operatorname{diag}\left(\bar{\boldsymbol{h}}_n^{r,k+1}\right) \boldsymbol{w}^{r,k+1}\right], \tag{50}$$

$$\lambda_{n,i}^{r,k+1} = \lambda_{n,i}^{r,k} + \rho h_{n,i}^{r,k+1}\left(w_{n,i}^{r,k+1} - w_i^{r,k+1}\right). \tag{51}$$

For analog communication, every i^{th} element of the updates from all agents is sent to the PS through a time-varying and noisy channel using the same subcarrier. In our simulation, the number of required time slots to transmit the devices' updates is proportional to the ratio between the size of the model d and the number of available subcarriers N_s, i.e., $\lceil \frac{d}{N_s} \rceil$. For example, for dataset *a9a* ($d = 123$), $\lceil \frac{123}{64} \rceil = 2$ time slots are required to upload the devices' updates.

For digital communication, every transmitted element is represented by 32 bits (full precession communication). Therefore, in contrast to the analog schemes, the required number of time slots depends on the number of bits to be sent, as well as the channel condition between the devices and the PS. In other words, the number of uploading time slots (τ_n) needed to send the updates from device n to the PS when the bandwidth is divided equally across devices is calculated as:

$$\sum_{t=0}^{\tau_n} \sum_{s=1}^{\frac{N_s}{N}} \tau R_{n,s}(t) \geq 32d, \tag{52}$$

where $R_{n,s}(t) = W \log_2(1 + P_n |h_{n,s}(t)|^2 / N_0 W)$ is the rate expression that follows from Shannon formula [38] and N_0 is the noise power spectral density which is set to 10^{-9}W/Hz. Since the devices have independent channel realizations, the required time to receive all the updates at the PS is $\bar{\tau} = \max\{\tau_1, \tau_2, \ldots, \tau_N\}$.

Baselines: Throughout the simulation section, we compare our proposed algorithms with respect to the following baselines

- *Newton-Zero:* Every device is required to compute and transmit the Hessian matrix only at the first iteration, whereas the gradient is computed and transmitted in all communication rounds following this simple update step

$$\boldsymbol{x}^{r+1} = \boldsymbol{x}^r - \frac{1}{N}\left[\sum_{n=1}^{N} \nabla^2 f_n\left(\boldsymbol{x}^0\right)\right]^{-1} \sum_{n=1}^{N} \nabla f_n(\boldsymbol{x}^r). \tag{53}$$

  This method only requires the second-order information at the first iteration, which is more communication-efficient compared to the standard Newton's method where the Hessian is computed, stored, and communicated at every communication round. Note that, in our proposed algorithm, we do not require transmitting the Hessian matrix at all. Due to the necessity of sharing the zeroth Hessian and the gradient, this algorithm may violate privacy.
- *FedGD:* The first-order FL based on distributed gradient descent algorithm which is common in solving federated optimization problems. The locally computed gradient at every device is transmitted at every communication round, then the PS updates the global model as follows

$$\boldsymbol{x}^{r+1} = \boldsymbol{x}^r - \alpha \frac{1}{N} \sum_{n=1}^{N} \nabla f_n(\boldsymbol{x}^r). \tag{54}$$

  Although it requires more rounds to converge, *FedGD* is still favored when the computation complexity is the bottleneck rather than communication resources. We chose the learning rate $\alpha = 10^{-3}$ since larger values cause the algorithm to diverge.
- *NDAM:* The digital version of our proposed formulation, when perfect channel estimation is assumed. Here, we divide the bandwidth equally between the devices so that the impact of limited bandwidth is measured for different datasets and network settings. We choose the same value

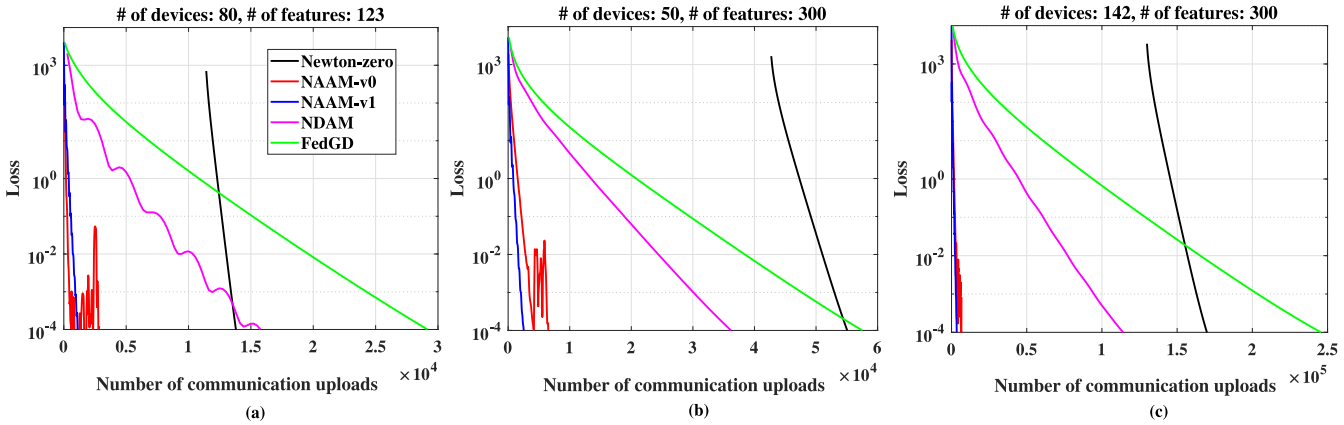


Fig. 3. Training loss versus the number of communication uploads: (a) *a9a* dataset, (b) *w7a* dataset, (c) *w8a* dataset.

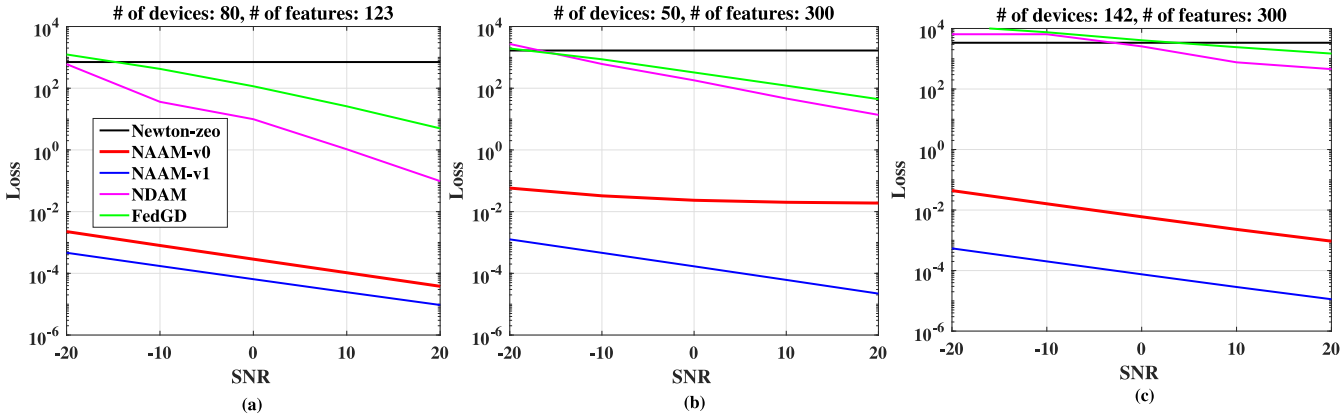


Fig. 4. Training loss versus SNR when the maximum number of channel uses is 10^4 ($CU = 10^4$): (a) *a9a* dataset, (b) *w7a* dataset, (c) *w8a* dataset.

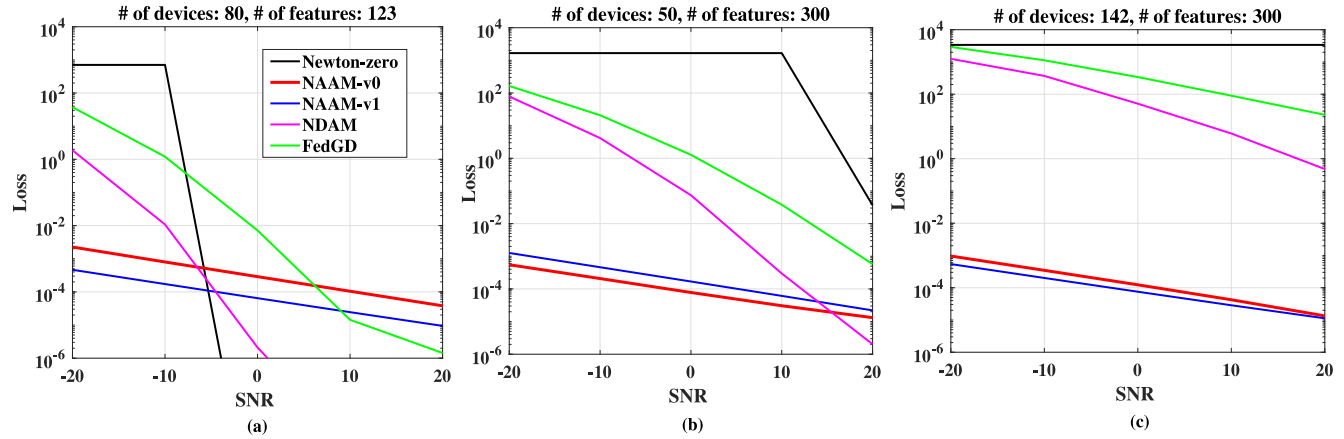


Fig. 5. Training loss versus SNR when the maximum number of channel uses is 5×10^4 ($CU = 5 \times 10^4$): (a) *a9a* dataset, (b) *w7a* dataset, (c) *w8a* dataset.

of ρ leading to fast convergence for all relevant algorithms under all settings.

A. Results and Discussion

The numerical results of our experiments are shown in Figs. 3-6. The performance of our proposed algorithms *NAAM-v0* and *NAAM-v1* (decribed in Section II) is measured with respect to different baselines using three datasets, as detailed in Table I. We plot the loss, $|f(\boldsymbol{x}^k) - f(\boldsymbol{x}^*)|$, with respect to different metrics. The optimal value $f(\boldsymbol{x}^*)$ for each setting is found by performing the standard Newton's method and getting the value at the convergence point.

In Fig. 3, we plot the training loss of the algorithms with respect to the number of communication uploads for different datasets and number of devices where communication uploads refer to the number of time slots that are needed for devices to upload their updates to the PS. We observe that both *NAAM-v0* and *NAAM-v1* require a very low number of communication uploads to reach the target training loss (10^{-4} in this case)

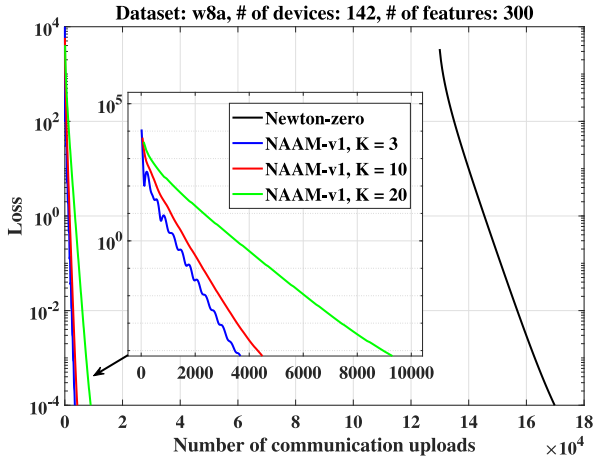


Fig. 6. Training loss versus the number of communication uploads for different number of ADMM steps $K = \{3, 10, 20\}$.

compared to *Newton-zero*, *FedGD*, and *NDAM* in all considered datasets and number of devices. For instance, in Fig. 3a, *Newton-zero*, *NDAM*, and *FedGD* require around 12, 14, and 26 times more number of communication uploads compared to *NAAM-v0*, respectively. This is justified by the adoption of analog *over-the-air aggregation* method where every i^{th} element from the devices can be transmitted using the same subcarrier, while the digital scheme implementation requires orthogonal subcarriers to transmit all the elements. We note that we chose $K = 3$ for *NAAM-v1* to get a good approximation of the second term in the right hand side of (53). For *NAAM-v0*, we select $\beta = 10^{-6}$ (channel inversion threshold) and $K = 10$ since it is the minimum number of local iterations that enables the algorithm to reach the target training loss. Nonetheless, *NAAM-v0* suffers from fluctuating performance at low training loss regime. This is due to the fact that channel inversion prevents devices from sending their elements when the channel is in deep fading. This limits the available information at the PS side, and thus results in an inaccurate update. Comparing the results for the different datasets, we notice that the number of transmitted elements greatly affects the digital communication baselines (*Newton-zero*, *FedGD*, and *NDAM*). For example, to send a model-size update, devices need to transmit $N \times d = \{9840, 15000, 42600\}$ elements for datasets *a9a*, *w7a*, and *a8a*, respectively. The results in Fig. 3 illustrate the fact that the larger the number of elements to be transmitted, the more communication uploads are required to reach a certain training loss. On the other hand, the performance of analog *over-the-air aggregation* baselines (*NAAM-v0* and *NAAM-v1*) is independent of the number of devices, which justifies the large gap with respect to the digital baselines.

Energy and Bandwidth Efficiency: Figs. 4-5 show the effects of available bandwidth on the training loss for different SNR values. To limit the bandwidth, we put a constraint on the maximum number of available channel uses $C_u = \sum_{j=1}^{J} S_j$, where S_j is the number of subcarriers available at time slot j. We consider different SNR regimes by changing the transmission power since $N_0 W$ is fixed. In Fig. 4, we limit the maximum number of channel uses to $CU = 10^4$. We see that the training loss achieved by *NAAM-v0* and *NAAM-v1* is much lower than the ones of digital scheme baselines, even at very low $-20\,dB$ SNR, i.e., low transmit power. As the SNR value increases, the performance of *NDAM* and *FedGD* improves and surpasses that of *Newton-zero*. The reason for this is that the better the SNR gets, the higher transmission rate is achieved, and less communication resources are needed to transmit the updates from the device. Hence, this helps *NDAM* and *FedGD* to run for more communication rounds and achieve better performance. On the other hand, despite the increase in SNR, by limiting the number of channel uses, high training loss is experienced by *Newton-zero*. This is due to the fact that the Hessian matrix is not fully transmitted, so the matrix is not complete which hinders the training accuracy. In Fig. 5, when we increase the maximum number of channel uses to $CU = 5 \times 10^4$, we notice in the case of dataset *a9a*, all digital baselines reach the target training loss at SNR $> 10\,dB$. Nonetheless, we see that the performance of the digital baselines degrades for other datasets with higher number of elements to be transmitted from each device since the competition on the available subcarriers increases. For dataset *w7a* in Fig. 5(b), *NAAM-v1* reaches the target loss at SNR $= 0\,dB$, whereas its digital counterpart *NDAM* requires SNR $= 12\,dB$ to reach the same target, and both *FedGD* and *Newton-zero* need SNR $> 20\,dB$. Hence, our algorithm provides more than 15 times reduction in transmission power, which validates it to be a highly energy-efficient solution. On the other hand, even at low SNR regimes for both choices of the number of channel uses, both analog schemes achieve low loss and significantly outperform the digital baselines. It is also worth mentioning that *NAAM-v1* (non-channel inversion based scheme) outperforms *NAAM-v0* (channel inversion based scheme).

Impact of number of local Iterations K: In previous simulation results, we assume running one inner iteration (one ADMM pass) at every outer iteration. In Fig. 6, we plot the training loss of *NAAM-v1* for different number of local iterations $K = \{3, 10, 20\}$ with respect to the number of communication uploads and compare it to *Newton-zero*. The more inner iterations, the more accurate the approximation of the model update $\boldsymbol{x}^r$ at every outer iteration r, which in turn comes at the cost of requiring more communication uploads (resources). Nonetheless, we clearly see that this additional overhead is not significant compared to the stringent requirements incurred by *Newton-zero*.

V. Conclusion

In this work, we presented a novel communication-efficient learning approach based on the second-order Newton-type method for solving a distributed FL problem. Our proposed algorithms *NAAM-v0* and *NAAM-v1* overcome the drawback of excessive communication resources when sending Hessian matrices from devices to the PS by allowing transmission of only one vector at each communication round. This is done by reformulating the Newton step as a solution to a convex quadratic problem and solving it using ADMM. Moreover, our approach ensures privacy since the transmitted

information by the devices does not expose the gradient nor the Hessian. Furthermore, our proposed framework enables us to leverage *analog over-the-air* model aggregation, which not only saves communication bandwidth but also adds another layer of privacy where the trajectory of updates from the devices is concealed by channel perturbations, as demonstrated for *NAAM-v1*. Our simulation results show that our proposed algorithms cope with noisy and time-varying channels and outperform the baselines with digital implementation in terms of required communication resources for different values of SNR. Results also show that *NAAM-v0* and *NAAM-v1* provide a robust and enhanced performance for bandwidth-limited systems, especially when the number of available devices becomes large. For future directions, this work can be extended to tackle modern ML problems that require deep neural network (DNN) structures which are highly stochastic and non-convex, in addition to addressing many challenges such as effective modulation and coding schemes, data heterogeneity at the devices, and synchronization between the devices and the PS.

REFERENCES

[1] M. Krouka, A. Elgabli, C. B. Issaid, and M. Bennis, "Energy-efficient model compression and splitting for collaborative inference over time-varying channels," in *Proc. IEEE 32nd Annu. Int. Symp. Pers. Indoor Mobile Radio Commun. (PIMRC)*, 2021, pp. 1173–1178.

[2] M. Krouka, A. Elgabli, C. B. Issaid, and M. Bennis, "Communication-efficient split learning based on analog communication and over the air aggregation," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2021, pp. 1–6.

[3] M. Krouka, A. Elgabli, and M. Bennis, "Maximum allowable transfer interval aware scheduling for wireless remote monitoring," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, 2020, pp. 1–6.

[4] E. Bastug, M. Bennis, and M. Debbah, "Living on the edge: The role of proactive caching in 5G wireless networks," *IEEE Commun. Mag.*, vol. 52, no. 8, pp. 82–89, Aug. 2014.

[5] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Stat.*, vol. 54, Apr. 2017, pp. 1273–1282.

[6] S. Oh, J. Park, E. Jeong, H. Kim, M. Bennis, and S.-L. Kim, "Mix2FLD: Downlink federated learning after uplink federated distillation with two-way mixup," *IEEE Commun. Lett.*, vol. 24, no. 10, pp. 2211–2215, Oct. 2020.

[7] J. D. Lee, Q. Lin, T. Ma, and T. Yang, "Distributed stochastic variance reduced gradient methods by sampling extra data with replacement," *J. Mach. Learn. Res.*, vol. 18, no. 122, pp. 1–43, 2017.

[8] Y. Arjevani and O. Shamir, "Communication complexity of distributed convex learning and optimization," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst. Vol. 1*, 2015, pp. 1756–1764.

[9] H. Yu, R. Jin, and S. Yang, "On the linear speedup analysis of communication efficient momentum SGD for distributed non-convex optimization," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 7184–7193.

[10] T. Chen, G. Giannakis, T. Sun, and W. Yin, "LAG: Lazily aggregated gradient for communication-efficient distributed learning," in *Advances in Neural Information Processing Systems*, vol. 31. Red Hook, NY, USA: Curran, 2018.

[11] T. Chen, K. Zhang, G. B. Giannakis, and T. Basar, "Communication-efficient policy gradient methods for distributed reinforcement learning," *IEEE Trans. Control Netw. Syst.*, early access, May 6, 2021, doi: 10.1109/TCNS.2021.3078100.

[12] A. Elgabli, J. Park, A. S. Bedi, C. B. Issaid, M. Bennis, and V. Aggarwal, "Q-GADMM: Quantized group ADMM for communication efficient decentralized machine learning," *IEEE Trans. Commun.*, vol. 69, no. 1, pp. 164–181, Jan. 2021.

[13] C. B. Issaid, A. Elgabli, J. Park, M. Bennis, and M. Debbah, "Communication efficient decentralized learning over bipartite graphs," *IEEE Trans. Wireless Commun.*, early access, Nov. 17, 2021, doi: 10.1109/TWC.2021.3126859.

[14] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu, "1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs," in *Proc. INTERSPEECH*, 2014, pp. 1058–1062.

[15] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar, "SignSGD: Compressed optimisation for non-convex problems," in *Proc. 35th Int. Conf. Mach. Learn.*, vol. 80, Jul. 2018, pp. 560–569.

[16] D. Alistarh, D. Grubic, J. Z. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient SGD via gradient quantization and encoding," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 1707–1718.

[17] A. T. Suresh, F. X. Yu, S. Kumar, and H. B. McMahan, "Distributed mean estimation with limited communication," in *Proc. 34th Int. Conf. Mach. Learn. Vol. 70*, 2017, pp. 3329–3337.

[18] J. Wangni, J. Wang, J. Liu, and T. Zhang, "Gradient sparsification for communication-efficient distributed optimization," in *Advances in Neural Information Processing Systems*, vol. 31. Red Hook, NY, USA: Curran, 2018.

[19] H. Wang, S. Sievert, S. Liu, Z. Charles, D. Papailiopoulos, and S. Wright, "ATOMO: Communication-efficient learning via atomic sparsification," in *Advances in Neural Information Processing Systems*, vol. 31. Red Hook, NY, USA: Curran Assoc., Inc., 2018.

[20] P. Han, S. Wang, and K. K. Leung, "Adaptive gradient sparsification for efficient federated learning: An online learning approach," in *Proc. IEEE 40th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2020, pp. 300–310.

[21] D. Alistarh, T. Hoefler, M. Johansson, S. Khirirat, N. Konstantinov, and C. Renggli, "The convergence of sparsified gradient methods," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 5977–5987.

[22] A. Beck, *Introduction to Nonlinear Optimization: Theory, Algorithms, and Applications With MATLAB*. Philadelphia, PA, USA: SIAM, 2014.

[23] C. Guille-Escuret, B. Goujaud, M. Girotti, and I. Mitliagkas, "A study of condition numbers for first-order optimization," in *Proc. AISTATS*, 2021, pp. 1261–1269.

[24] X. Yin, B. W.-H. Ng, J. He, Y. Zhang, and D. Abbott, "Accurate image analysis of the retina using hessian matrix and binarisation of thresholded entropy with application of texture mapping," *PLoS One*, vol. 9, no. 4, pp. 1–17, Apr. 2014.

[25] R. H. Byrd, S. L. Hansen, J. Nocedal, and Y. Singer, "A stochastic quasi-Newton method for large-scale optimization," *SIAM J. Optim.*, vol. 26, no. 2, pp. 1008–1031, 2016.

[26] N. N. Schraudolph, J. Yu, and S. Günter, "A stochastic quasi-Newton method for online convex optimization," in *Proc. Int. Conf. Artif. Intell. Stat.*, 2007, pp. 436–443.

[27] P. Moritz, R. Nishihara, and M. Jordan, "A linearly-convergent stochastic L-BFGS algorithm," in *Proc. Int. Conf. Artif. Intell. Stat.*, 2016, pp. 249–258.

[28] C. Dünner, A. Lucchi, M. Gargiani, A. Bian, T. Hofmann, and M. Jaggi, "A distributed second-order algorithm you can trust," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1358–1366.

[29] S. Wang, F. Roosta, P. Xu, and M. W. Mahoney, "GIANT: Globally improved approximate Newton method for distributed optimization," in *Advances in Neural Information Processing Systems*, vol. 31. Red Hook, NY, USA: Curran, 2018.

[30] O. Shamir, N. Srebro, and T. Zhang, "Communication-efficient distributed optimization using an approximate Newton-type method," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 1000–1008.

[31] M. Safaryan, R. Islamov, X. Qian, and P. Richtarik, "FedNL: Making Newton-type methods applicable to federated learning," 2021, *arXiv:2106.02969*.

[32] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning via over-the-air computation," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 2022–2035, Mar. 2020.

[33] M. M. Amiri and D. Gündüz, "Machine learning at the wireless edge: Distributed stochastic gradient descent over-the-air," *IEEE Trans. Signal Process.*, vol. 68, pp. 2155–2169, Mar. 2020.

[34] A. Elgabli, J. Park, C. B. Issaid, and M. Bennis, "Harnessing wireless channels for scalable and privacy-preserving federated learning," *IEEE Trans. Commun.*, vol. 69, no. 8, pp. 5194–5208, Aug. 2021.

[35] M. Krouka, A. Elgabli, C. B. Issaid, and M. Bennis, "Communication-efficient and federated multi-agent reinforcement learning," *IEEE Trans. Cogn. Commun. Netw.*, vol. 8, no. 1, pp. 311–320, Mar. 2022.

[36] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 1–27, May 2011.

[37] "TS 38.211 v15.2.0 release 15tr 38.802 v14.1.0," 3GPP, Sophia Antipolis, France, Rep. DTS/TSGR-0138211vf20, Jun. 2017.

[38] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. New York, NY, USA: Cambridge Univ. Press, 2005.

Mounssif Krouka received the B.Sc. degree from the Institute of Electrical and Electronics Engineering (IGEE ex-INELEC), Boumerdes University, Algeria, in 2015, and the M.Sc. degree in wireless communications engineering from the University of Oulu, Oulu, Finland, in 2018, where he is a Doctoral Researcher with the Center for Wireless Communications. His research interests include ultra-reliable low-latency communications, distributed optimization, resource scheduling, and machine learning.

Anis Elgabli (Member, IEEE) received the B.Sc. degree in electrical and electronic engineering from the University of Tripoli, Libya, in 2004, the M.Eng. degree from UKM, Malaysia, in 2007, and the M.Sc. and Ph.D. degrees from the Department of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA, in 2015 and 2018, respectively. He is currently a Postdoctoral Researcher with the Centre for Wireless Communications, University of Oulu. His main research interests include heterogeneous networks, radio resource management, vehicular communications, video streaming, and distributed machine learning. He was a recipient of the Best Paper Award in HotSpot Workshop in 2018 (Infocom 2018) and the most JUFO points in 2020 at the Center of Wireless Communication, University of Oulu.

Chaouki Ben Issaid (Member, IEEE) received the Diplôme d'Ingénieur degree in economics and financial engineering from the Ecole Polytechnique de Tunisie in 2013, and the master's degree in applied mathematics and computational science and the Ph.D. degree in statistics from the King Abdullah University of Science and Technology in 2015 and 2019, respectively. He is currently a Postdoctoral Fellow with the Centre for Wireless Communications, University of Oulu. His current research interests include communication-efficient distributed learning and machine learning applications for wireless communication.

Mehdi Bennis (Fellow, IEEE) is a Professor with the Centre for Wireless Communications and a Academy of Finland Research Fellow and the Head of the Intelligent Connectivity and Networks/Systems Group (ICON), University of Oulu, Finland. He has published more than 200 research papers in international conferences, journals and book chapters. His main research interests are in radio resource management, heterogeneous networks, game theory and distributed machine learning in 5G networks, and beyond. He was a recipient of several prestigious awards, including the 2015 Fred W. Ellersick Prize from the IEEE Communications Society, the 2016 Best Tutorial Prize from the IEEE Communications Society, the 2017 EURASIP Best paper Award for the *Journal of Wireless Communications and Networks*, the all-University of Oulu award for research, the 2019 IEEE ComSoc Radio Communications Committee Early Achievement Award, and the 2020 Clarviate Highly Cited Researcher by the Web of Science. He is an Editor of IEEE TRANSACTIONS ON COMMUNICATIONS and a Specialty Chief Editor for Data Science for Communications in the Frontiers in Communications and Networks journal.