Communication-Efficient Second-Order Newton-Type Approach for Decentralized Learning

Mounssif Krouka, Anis Elgabli, Chaouki ben Issaid, and Mehdi Bennis
Centre for Wireless Communications (CWC)
University of Oulu, Finland
Email: {mounssif.krouka, anis.elgabli, chaouki.benissaid, mehdi.bennis}@oulu.fi

***Abstract*—In this paper, we propose a decentralized Newton-type approach to solve the problem of decentralized federated learning (FL). Notably, our proposed algorithm leverages the fast convergence of the second-order methods while avoid sending the hessian matrix at each iteration. Therefore, the proposed approach significantly reduces the communication cost and preserves the privacy. Specifically, we alternate between two problems. The inner problem approximates the inverse Hessian-gradient product which is formulated as a quadratic optimization problem and approximately solved in a decentralized manner using one step of the group alternating direction method of multipliers (GADMM) method. The outer problem learns the model, which is solved by performing one decentralized Newton step at every iteration. Moreover, to reduce the communication-overhead per iteration, a quantized version (leveraging stochastic quantization) is also proposed. Simulation results illustrate that our algorithm outperforms the baselines of GADMM, Q-GADMM, Newton tracking, and Decentralized SGD, and provides energy and communication-efficient solutions for bandwidth-limited systems under different SNR regimes.**

***Index Terms*—Decentralized optimization, communication-efficient Federated learning, decentralized Newton method, model quantization.**

I. Introduction

The rapid development in wireless technologies has led to the prevalent connectivity of a massive number of devices to the internet, generating huge amounts of data. This abundance of information allowed the emergence of highly performant machine learning (ML) algorithms that are applied in many scenarios, such as trajectory optimization [1], image classification [2], and resource scheduling [3]. The conventional way to train such ML algorithms is to send raw information from the devices/nodes to a powerful remote parameter server (PS), where all the computational processes occur. However, sharing raw data creates privacy and communication issues. i.e., huge communication resources are required, which can be a bottleneck for bandwidth-limited systems. Consequently, federated learning (FL) has recently gained attention due to its ability to enable privacy-preserving and communication-efficient training. In FL, instead of exchanging raw data, the nodes only transmit small-sized updates such as the model or gradient, which provides a significant reduction in communication cost. Also, the system privacy is preserved since the raw data is kept locally at every node. The state-of-the-art algorithms in FL utilize first-order gradient-based methods and propose ways to make it more communication-efficient. However, recently, second-order based schemes for the PS-based setting have been investigated. In this paper, we propose a communication-efficient and fully decentralized algorithm that is based on the second-order method. In what follows, we discuss the main aspects of these works and highlight the contributions of our proposed algorithm.

This work is supported by the European Union's Horizon Europe program through the project ADROIT6G, CENTRIC, and DESIRE6G.

A. First-order Methods

FL is commonly performed using first-order methods. For example, when using local SGD, at each iteration, the participating nodes solve the learning task by locally updating the global model using one/few gradient descent steps and then sending the updated models to the PS, which averages them to produce the new global model. However, the exchange between the PS and the nodes requires a large number of communication rounds until convergence, hence, incurring high latency and communication costs. For this reason, many works proposed communication-efficient solutions that i) reduce the payload size per communication round by applying different tools such as quantization [4], sparsification [5], [6], and gradient reuse [7], and ii) reduce the number of communication rounds by using techniques including adaptive learning rates [8], [9] and momentum acceleration [10].

B. Second-order Methods

Despite their popularity and ease of implementation, first-order methods suffer from slow convergence due to their dependence on the *condition number*, where the latter is related to the convexity and the smoothness of the loss function. To overcome this issue, second-order methods were proposed. By leveraging second-order information, the convergence speed can be accelerated by utilizing both the Hessian and the gradient in the learning process. The standard Newton's method at training iteration $r+1$ can be written as follows

$$\boldsymbol{x}^{r+1} = \boldsymbol{x}^{r} - (\nabla^2 f(\boldsymbol{x}^{r}))^{-1} \nabla f(\boldsymbol{x}^{r}), \tag{1}$$

where $\boldsymbol{x}^r$, $\nabla f(\boldsymbol{x}^r)$, and $\nabla^2 f(\boldsymbol{x}^r)$ are the model, gradient and Hessian of $f(\boldsymbol{x}^r)$, respectively. The solution to this problem requires all the nodes to compute the gradient and the Hessian locally and then transmit them to the PS. Nonetheless, this

incurs huge communication costs. To tackle this issue, the work in [11] suggested a communication-efficient distributed asynchronous quasi-Newton algorithm where nodes transmit three model-sized vectors to the PS. Moreover, the authors in [12] proposed *FedNL*, a communication-efficient Hessian learning technique, where the nodes share a compressed version of the Hessian to the PS. Although this technique reduces the communication cost, the compression operation depends on the rank of the Hessian matrix. The same article suggested another algorithm called *Newton-zero*, where the Hessian matrix is computed and transmitted only at the first learning iteration $r = 0$, as shown below

$$\boldsymbol{x}^{r+1} = \boldsymbol{x}^{r} - (\nabla^2 f(\boldsymbol{x}^0))^{-1} \nabla f(\boldsymbol{x}^r). \tag{2}$$

We note that computing the Hessian only at the first iteration minimizes the local computational complexity. However, transmitting the zero Hessian matrix requires more communication resources than model-sized vector sharing, especially when the model's size is large.

C. Decentralized Approaches

Note that the above-mentioned algorithms require the PS to collect the information from all the nodes. This scheme may not be communication-efficient as: i) the competition over the communication resources increases when more nodes participate in the learning, which creates a bottleneck for limited bandwidth systems, ii) the nodes can be dispersed and located outside the PS coverage zone, and iii) the communication rate is hindered by the node experiencing the weakest communication channel. Accordingly, works on decentralized solutions for both first-order [13]–[15] and second-order [16]–[18] methods have been investigated. For first-order decentralized methods, the work in [13] presented a generalized decentralized stochastic gradient descent (SGD) framework for gradient sharing. On the other hand, the authors in [14] considered a framework named *GADMM*, based on the alternating direction method of multipliers (ADMM), where each node is communicating with only two neighbors, and at most half of the workers are competing on the limited communication resources at any given time. A quantized version of *GADMM* was also proposed in [15]. Regarding second-order decentralized methods, the authors in [16] proposed a decentralized consensus algorithm using a quadratic approximation of the objective function, where each node updates its local variable using neighboring and historical information. The work in [17] considered a distributed adaptive Newton algorithm with a global quadratic convergence rate that requires $N-1$ rounds of information exchange at every communication round. Nonetheless, combining both communication efficiency of first-order methods and fast convergence of second-order methods can further improve the communication cost.

D. Contributions & Paper Organization

This paper proposes a novel bilevel decentralized second-order learning approach that attains the fast convergence property of second-order methods and low communication cost per iteration of first-order methods. In particular, for the inner level problem, we formulate the learning of the *inverse zero Hessian-gradient product* as a quadratic optimization problem, solved in a decentralized manner. Precisely, we approximate the solution by applying one *GADMM* step [14], where every node sends only two model-sized vector updates to its two neighbors, which reduces the required communication resources. Next, the model is updated by performing one Newton step at every iteration. Finally, we leverage stochastic quantization to achieve more communication-efficient model training. The major contributions of our work are summarized as follows.

- To extend our previous work in [19], we propose a second-order Newton-type approach to solve the problem of decentralized FL. In particular, we Leverage the *GADMM* algorithm to provide a communication-efficient approach, coined as *DNGAM*–**D**ecentralized **N**ewton **G**roup **A**DM**M**, that enables the inner-level problem solution approximation in a decentralized manner.
- We apply stochastic quantization following [15] to further reduce the communication cost.
- Simulation results illustrate that our algorithm *Q-DNGAM* outperforms the baselines and provides a energy and communication-efficient solution. Additionally, our algorithm shows a robust and enhanced performance for bandwidth-limited systems under different SNR regimes.

The rest of the paper is structured as follows. In Section II, we describe the system model and formulate our proposed algorithm *DNGAM*. Then, in Section II-C, we illustrate the steps for *Q-DNGAM* as we apply the quantization operation for further reduction in communication cost. In Section III, we present and discuss the simulation results. Finally, we conclude our work in Section IV.

II. Problem Formulation and Proposed Algorithm

We consider a system consisting of N nodes, each having $f_n(\boldsymbol{x})$ as a local loss function evaluated at the model $\boldsymbol{x} \in \mathbb{R}^d$, where d is the size of the model. Our aim is to solve the following learning problem in a fully decentralized way

$$\min_{\boldsymbol{x}\in\mathbb{R}^d} \; f(\boldsymbol{x}) \;\; \text{with} \;\; f(\boldsymbol{x}) = \frac{1}{N}\sum_{n=1}^{N} f_n(\boldsymbol{x}). \tag{3}$$

If problem (3) is solved using *Newton-zero* with the aid of a PS, then the updating step at iteration $r+1$ will be:

$$\boldsymbol{x}^{r+1} = \boldsymbol{x}^{r} - \Big(\frac{1}{N}\sum_{n=1}^{N} \nabla^2 f_n(\boldsymbol{x}^0)\Big)^{-1} \Big(\frac{1}{N}\sum_{n=1}^{N} \nabla f_n(\boldsymbol{x}^r)\Big), \tag{4}$$

where $\nabla^2 f_n(\boldsymbol{x}^0) \in \mathbb{R}^{d\times d}$ and $\nabla f_n(\boldsymbol{x^r})$ are the zero Hessian (i.e., Hessian at $r = 0$) and the gradient of $f_n(\boldsymbol{x}^r)$, respectively. For ease of notation, we use $\boldsymbol{H}_n^0 = \nabla^2 f_n(\boldsymbol{x}^0)$ and $\boldsymbol{g}_n^r = \nabla f_n(\boldsymbol{x}^r)$. Moreover, we define the system Hessian and gradient as

$$\boldsymbol{H}^0 = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{H}_n^0 \quad \text{and} \quad \boldsymbol{g}^r = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{g}_n^r. \tag{5}$$

Hence, we can write the step in (4) as follows

$$\boldsymbol{x}^{r+1} = \boldsymbol{x}^r - (\boldsymbol{H}^0)^{-1}\boldsymbol{g}^r. \tag{6}$$

We note that the second term on the right hand side of (6) is computed at every iteration r. Equivalently, we can replace the *inverse zero Hessian-gradient product* with the exact solution of the following optimization problem

$$\boldsymbol{w}^{r\star} = \underset{\boldsymbol{w}\in\mathbb{R}^d}{\arg\min}\ \frac{1}{2}\boldsymbol{w}^T\boldsymbol{H}^0\boldsymbol{w} - \boldsymbol{w}^T\boldsymbol{g}^r. \tag{7}$$

Specifically, $\boldsymbol{w}^{r\star} = (\boldsymbol{H}^0)^{-1}\boldsymbol{g}^r$ is the optimal solution to problem (7). Nevertheless, the solution of this problem cannot be obtained in a decentralized manner. To this end, we reformulate the problem in (7) as the following decentralized problem,

$$\begin{aligned}\boldsymbol{w}^{r\star} = \underset{\{\boldsymbol{w}_n\}_{n=1}^N\in\mathbb{R}^d}{\arg\min}\ &\sum_{n=1}^N \frac{1}{2}\boldsymbol{w}_n^T\boldsymbol{H}_n^0\boldsymbol{w}_n - \boldsymbol{w}_n^T\boldsymbol{g}_n^r\\ s.t.\ \ &\boldsymbol{w}_n = \boldsymbol{w}_{n+1},\ \ \forall\ n = \{1,2,\cdots,N-1\}\end{aligned} \tag{8}$$

where $\boldsymbol{w}^{r\star}$ is the optimal solution. At the convergence point, the constraint in (8) implies that $\boldsymbol{w}_{n-1}^{r\star} = \boldsymbol{w}_n^{r\star}$ and $\boldsymbol{w}_n^{r\star} = \boldsymbol{w}_{n+1}^{r\star}$ for all n. Hence, a unique global update $\boldsymbol{w}^{r\star}$ is obtained by all the nodes. However, solving (8) at every iteration r to update the model $\boldsymbol{x}^{r+1}$ can also be expensive in terms of the communication cost since it introduces a double loop with many communication rounds. Hence, we propose to perform one update for the solution of the inner problem at each outer (Newton) step. In particular, We approximate the *inverse zero Hessian-gradient product*, i.e, the solution of (8), by performing one *GADMM* iteration. Then, we apply a Newton step to update the model $\boldsymbol{x}^{r+1}$ for all the nodes. In the following, we explain the details of the proposed framework to solve the problem in (3).

A. Decentralized Newton Group ADMM

As previously mentioned, the idea behind the *DNGAM* algorithm is to cluster the nodes into two groups, namely *head* and *tail*. Thus, at every communication round, only half the nodes are sending their two vector updates to their two corresponding neighbors from the other group (except for the first and the last nodes). Let $\mathcal{N}_h = \{1,3,\ldots,N-1\}$ and $\mathcal{N}_t = \{2,4,\ldots,N\}$ denote the *head* and *tail* groups, respectively. We formulate the augmented Lagrangian for problem (8) as

$$\begin{aligned}&\mathcal{L}_\rho(\{\boldsymbol{w}_n\}_{n=1}^N, \boldsymbol{\lambda})\\ &= \sum_{n=1}^N \big[\frac{1}{2}\boldsymbol{w}_n^T\boldsymbol{H}_n^0\boldsymbol{w}_n - \boldsymbol{w}_n^T\boldsymbol{g}_n^r\big] + \sum_{n=1}^N \langle\boldsymbol{\lambda}_n, \boldsymbol{w}_n - \boldsymbol{w}_{n+1}\rangle\\ &+ \frac{\rho}{2}\sum_{n=1}^N \left\|\boldsymbol{w}_n - \boldsymbol{w}_{n+1}\right\|^2,\end{aligned} \tag{9}$$

where $\boldsymbol{\lambda}_n$ is the dual variable for node n and $\rho > 0$ is the parameter that adjusts the mismatch between the primal variables of nodes n and $n+1$. At iteration $r+1$, every node n in the *head* group updates the primal variable $\boldsymbol{w}_n^{r+1}$ by solving

$$\boldsymbol{w}_n^{r+1} = \underset{\boldsymbol{w}_n}{\arg\min}\ \mathcal{L}_\rho(\{\boldsymbol{w}_n\}_{n=1}^N, \boldsymbol{\lambda}_n^r). \tag{10}$$

Taking the derivative of $\mathcal{L}_\rho(\{\boldsymbol{w}_n\}_{n=1}^N, \boldsymbol{\lambda}_n^r)$ and equating to zero, yields the following closed-form expression

$$\boldsymbol{w}_n^{r+1} = \big(\boldsymbol{H}_n^0 + 2\rho\boldsymbol{I}\big)^{-1}\big(\boldsymbol{g}_n^r + \boldsymbol{\lambda}_{n-1}^r - \boldsymbol{\lambda}_n^r + \rho(\boldsymbol{w}_{n-1}^r + \boldsymbol{w}_{n+1}^r)\big). \tag{11}$$

After that, the local version of the model $\boldsymbol{x}_n^{r+\frac{1}{2}}$ is computed

$$\boldsymbol{x}_n^{r+\frac{1}{2}} = \boldsymbol{x}_n^r - \alpha\boldsymbol{w}_n^{r+1}, \tag{12}$$

where α is a step size we introduce. In the subsequent communication round, after receiving $\boldsymbol{w}_{n\in\mathcal{N}_h}^{r+1}$ and $\boldsymbol{x}_{n\in\mathcal{N}_h}^{r+\frac{1}{2}}$, each node n in the *tail* group computes the primal variable $\boldsymbol{w}_n^{r+1}$ as

$$\boldsymbol{w}_n^{r+1} = \big(\boldsymbol{H}_n^0 + 2\rho\boldsymbol{I}\big)^{-1}\big(\boldsymbol{g}_n^r + \boldsymbol{\lambda}_{n-1}^r - \boldsymbol{\lambda}_n^r + \rho(\boldsymbol{w}_{n-1}^{r+1} + \boldsymbol{w}_{n+1}^{r+1})\big), \tag{13}$$

and the local model $\boldsymbol{x}_n^{r+\frac{1}{2}}$ using (12), followed by a consensus step where each node in the *tail* group averages its local model $\boldsymbol{x}_n^{r+\frac{1}{2}}$ with its two neighbors from the *head* group, as shown below

$$\boldsymbol{x}_n^{r+1} = \frac{1}{3}(\boldsymbol{x}_{n-1}^{r+\frac{1}{2}} + \boldsymbol{x}_n^{r+\frac{1}{2}} + \boldsymbol{x}_{n+1}^{r+\frac{1}{2}}). \tag{14}$$

Next, after receiving the updates from *tail* neighbors ($\boldsymbol{w}_{n\in\mathcal{N}_t}^{r+1}$ and $\boldsymbol{x}_{n\in\mathcal{N}_t}^{r+1}$), each node in the *head* group updates the model $\boldsymbol{x}_n^{r+1}$ using (14). Finally, every node $n \in \mathcal{N}$ updates the dual variables $\boldsymbol{\lambda}_{n-1}^{r+1}$ and $\boldsymbol{\lambda}_n^{r+1}$ using

$$\boldsymbol{\lambda}_n^{r+1} = \boldsymbol{\lambda}_n^r - \rho(\boldsymbol{w}_n^{r+1} - \boldsymbol{w}_{n+1}^{r+1}), \tag{15}$$

The detailed steps of *DNGAM* are shown in Algorithm 1. As mentioned earlier, to perform one *DNGAM* iteration, every node has to transmit two vectors to its two neighboring nodes: (i) the primal update from the approximate solution of (8), and (ii) the model $\boldsymbol{x}_n$. In the next section, we consider applying stochastic quantization to reduce the communication overhead per iteration, i.e., the quantized versions of the updates from the neighbors are used to update the variables of the inner problem (primal and dual) and the model, for every node n.

B. Quantization Operation

In this subsection, we describe the quantization process following the work in [15]. For the sake of brevity, we describe the quantization operations only for the primal variable since the same steps apply to the model quantization.

At iteration r, every node n quantizes the difference between the current primal variable and the previous quantized primal variable such that $\boldsymbol{w}_n^r - \hat{\boldsymbol{w}}_n^{r-1} = Q_n(\boldsymbol{w}_n^r, \hat{\boldsymbol{w}}_n^{r-1})$, where $Q_n(.)$ is the stochastic quantization operation that depends on both the number of bits b_n^r and the quantization probability $p_{n,i}^r$ for every vector element i. We place the i^{th} element $[\hat{\boldsymbol{w}}_n^{r-1}]_i$ of the previously quantized variable at the center of the quantization range $2R_n^r$ which is equally divided into $2^{b_n^r} - 1$ levels. Hence, the quantization step size is $\delta_n^r = 2R_n^r/(2^{b_n^r} - 1)$. Consequently, the difference between

Algorithm 1 DNGAM

Input: $N, f_n(\cdot), \rho, R, d, \quad \forall n,$
Output: $\boldsymbol{x} \quad \forall n$
Initialization: $\boldsymbol{x}_n^0, \boldsymbol{w}_n^0, \boldsymbol{\lambda}_n^0, \quad \forall n.$
while $r \leq R$ **do**
  ***Head* group nodes ($n \in \mathcal{N}_h$): in parallel**
  Compute $\boldsymbol{w}_n^{r+1}$ via (11) then update $\boldsymbol{x}_n^{r+\frac{1}{2}}$ via (12)
  Send $\boldsymbol{w}_n^{r+1}$, $\boldsymbol{x}_n^{r+\frac{1}{2}}$ to the neighbors $n-1$ and $n+1$
  ***Tail* group nodes ($n \in \mathcal{N}_t$): in parallel**
  Compute $\boldsymbol{w}_n^{r+1}$ via (13), $\boldsymbol{x}_n^{r+\frac{1}{2}}$ via (12), and $\boldsymbol{x}_n^{r+1}$ via (14)
  Send $\boldsymbol{w}_n^{r+1}$, $\boldsymbol{x}_n^{r+1}$ to the neighbors $n-1$ and $n+1$
  Every node updates $\boldsymbol{\lambda}_{n-1}^{r+1}$ and $\boldsymbol{\lambda}_n^{r+1}$ via (15) and $\boldsymbol{x}_{n\in\mathcal{N}_h}^{r+1}$ via (14)
end while

the i^{th} element of the current update $[\boldsymbol{w}_n^r]_i$ and $[\hat{\boldsymbol{w}}_n^{r-1}]_i$ is computed as

$$[c_n(\boldsymbol{w}_n^r)]_i = \frac{1}{\delta_n^r}\Big([\boldsymbol{w}_n^r]_i - [\hat{\boldsymbol{w}}_n^{r-1}]_i + R_n^r\Big), \tag{16}$$

where we add the term R_n^r to ensure that the quantized value is non-negative. Hence, we map $[c_n(\boldsymbol{w}_n^r)]_i$ to the quantization levels as follows

$$[q_n(\boldsymbol{w}_n^r)]_i = \begin{cases} \lceil [c_n(\boldsymbol{w}_n^r)]_i \rceil & \text{with probability } p_n^r \\ \lfloor [c_n(\boldsymbol{w}_n^r)]_i \rfloor & \text{with probability } 1-p_n^r \end{cases} \tag{17}$$

where $\lceil\cdot\rceil$ and $\lfloor\cdot\rfloor$ stand for the ceiling and floor functions, respectively. According to [15], to ensure an unbiased quantization error, and to guarantee non-increasing quantization step sizes which is required for algorithm convergence, the choices of both p_n^r and b_n^r are given, respectively, as follows

$$p_n^r = \Big([c_n(\boldsymbol{w}_n^r)]_i - \lfloor [c_n(\boldsymbol{w}_n^r)]_i \rfloor\Big), \tag{18}$$

$$b_n^r \geq \left\lceil \log_2\left(1 + (2^{b_n^{r-1}} - 1)R_n^r / R_n^{r-1}\right)\right\rceil. \tag{19}$$

As a result, for node n to reconstruct the quantization variable $\hat{\boldsymbol{w}}_n^r$, it requires the quantities R_n^r, b_n^r, and $q_n(\boldsymbol{w}_n^r)$ to be received from its two neighboring nodes. The reconstruction step is done as follows

$$\hat{\boldsymbol{w}}_n^r = \hat{\boldsymbol{w}}_n^{r-1} + \delta_n^r q_n^r(\boldsymbol{w}_n^r) - R_n^r \mathbf{1}. \tag{20}$$

The required number of bits to transmit one vector update at every communication round using *Q-DNGAM* is $b_n^r d + (b_n^r + R_n^r)$ where $R_n^r \leq 32$ and $b_n^r \leq 32$, compared to $32d$ bits for *DNGAM*. This results in communication-efficient solution with a huge reduction in communication resources.

C. Q-DNGAM Updates

Here, we detail the update steps for *Q-DNGAM* to solve the problem (3). First, every node n in the *head* group updates its primal variable $\boldsymbol{w}_n^{r+1}$ as follows

$$\boldsymbol{w}_n^{r+1} = \big(\boldsymbol{H}_n^0 + 2\rho\boldsymbol{I}\big)^{-1}\big(\boldsymbol{g}_n^r + \boldsymbol{\lambda}_{n-1}^r - \boldsymbol{\lambda}_n^r + \rho(\hat{\boldsymbol{w}}_{n-1}^r + \hat{\boldsymbol{w}}_{n+1}^r)\big), \quad n \in \mathcal{N}_h \setminus \{1\}. \tag{21}$$

After that, the local model $\boldsymbol{x}_n^{r+\frac{1}{2}}$ is updated using (12). Subsequently, every *head* node transmits the quantized versions of its primal variable $\hat{\boldsymbol{w}}_n^{r+1}$ and local model $\hat{\boldsymbol{x}}_n^{r+\frac{1}{2}}$ to its *tail* neighbors, and every *tail* node updates its primal variable as

$$\boldsymbol{w}_n^{r+1} = \big(\boldsymbol{H}_n^0 + 2\rho\boldsymbol{I}\big)^{-1}\big(\boldsymbol{g}_n^r + \boldsymbol{\lambda}_{n-1}^r - \boldsymbol{\lambda}_n^r + \rho(\hat{\boldsymbol{w}}_{n-1}^{r+1} + \hat{\boldsymbol{w}}_{n+1}^{r+1})\big), \quad n \in \mathcal{N}_t \setminus \{N\}. \tag{22}$$

Moreover, the model is locally updated using (12) and then averaged with the quantized local models from the *head* neighbors, such that

$$\boldsymbol{x}_n^{r+1} = \frac{1}{3}(\boldsymbol{x}_n^{r+\frac{1}{2}} + \hat{\boldsymbol{x}}_{n-1}^{r+\frac{1}{2}} + \hat{\boldsymbol{x}}_{n+1}^{r+\frac{1}{2}}), \quad n \in \mathcal{N} \setminus \{1, N\}. \tag{23}$$

Next, the *head* nodes receive $\hat{\boldsymbol{w}}_{n\in\mathcal{N}_t}^{r+1}$ and $\hat{\boldsymbol{x}}_{n\in\mathcal{N}_t}^{r+1}$ from *tail* neighbors and update $\hat{\boldsymbol{x}}_n^{r+1}$ using (23). Finally, all the nodes $n \in \mathcal{N}$ update their dual variables as follows

$$\boldsymbol{\lambda}_n^{r+1} = \boldsymbol{\lambda}_n^r - \rho(\boldsymbol{w}_n^{r+1} - \hat{\boldsymbol{w}}_{n+1}^{r+1}). \tag{24}$$

Note that node $n=1$ communicates solely with node $n=2$. Hence, its update steps for $\boldsymbol{w}_1^{r+1}$ and $\boldsymbol{x}_1^{r+1}$ are

$$\boldsymbol{w}_1^{r+1} = \big(\boldsymbol{H}_1^0 + 2\rho\boldsymbol{I}\big)^{-1}\big(\boldsymbol{g}_1^r - \boldsymbol{\lambda}_1^r + \rho\hat{\boldsymbol{w}}_2^r\big), \tag{25}$$

$$\boldsymbol{x}_1^{r+1} = \frac{1}{2}(\boldsymbol{x}_1^{r+\frac{1}{2}} + \hat{\boldsymbol{x}}_2^{r+1}). \tag{26}$$

Similarly, node $n=N$ can only exchange the updates with node $n=N-1$. Thus, we get the following update expressions

$$\boldsymbol{w}_N^{r+1} = \big(\boldsymbol{H}_N^0 - 2\rho\boldsymbol{I}\big)^{-1}\big(\boldsymbol{g}_N^r + \boldsymbol{\lambda}_{N-1}^r + \rho\hat{\boldsymbol{w}}_{N-1}^r\big), \tag{27}$$

$$\boldsymbol{x}_N^{r+1} = \frac{1}{2}(\boldsymbol{x}_N^{r+\frac{1}{2}} + \hat{\boldsymbol{x}}_{N-1}^{r+\frac{1}{2}}). \tag{28}$$

III. Simulation Results

In this section, we numerically evaluate the performance of *DNGAM* and *Q-DNGAM* apropos different baselines: *GADMM*, *Q-GADMM*, *Newton tracking*, and *DSGD*. We select $\alpha = 0.13$ and $\rho = 500$. In the following, we describe the learning problem, the network and communication settings, and the simulation results.

A. Simulation Setting & Baselines

We consider the regularized logistic regression problem

$$\min_{x\in\mathcal{R}^d} \Big\{f(x) := \frac{1}{N}\sum_{n=1}^{N} f_n(x) + \frac{\eta}{2}||x||^2\Big\}, \tag{29}$$

where

$$f_n(x) = \frac{1}{M}\sum_{m=1}^{M} \log\big(1 + \exp(-b_{n,m}a_{n,m}^T x)\big). \tag{30}$$

Where $\{a_{n,m}, b_{n,m}\}_{m\in[M]}$ denote the data samples for node n, M is the number of samples allocated for every node, and η is a regularization parameter. We make use of the dataset *a1a* from LibSVM [20].

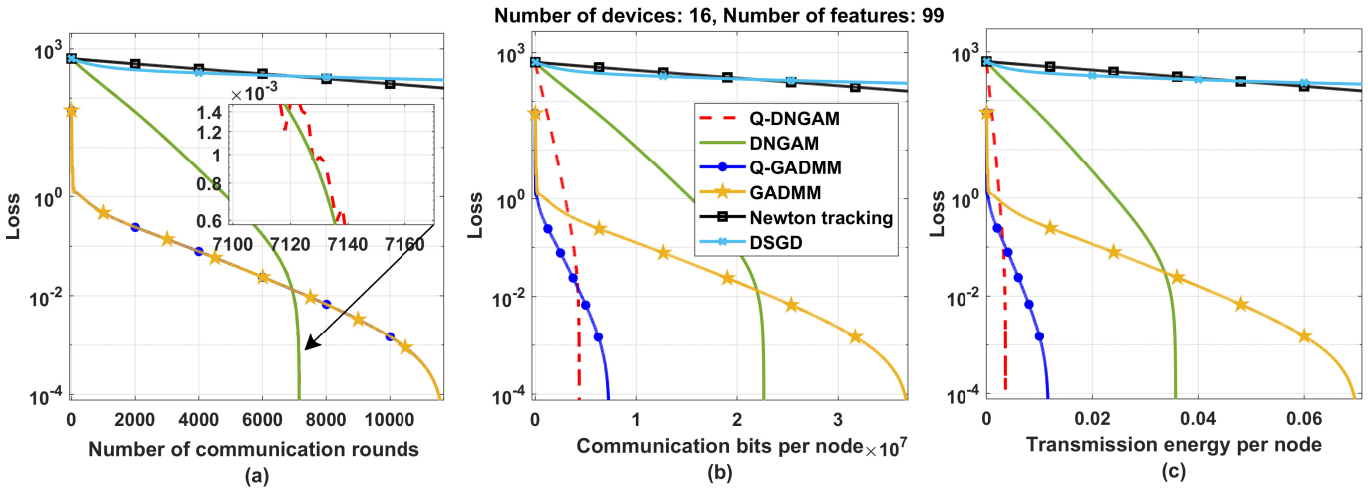

Fig. 1: Loss with respect to: a) number of communication rounds, b) communication bits per node, and c) transmission energy per node (Joule).

For all the simulation results, we set a fixed number of subcarriers $N_s = 64$ divided across all the nodes, where every subcarrier provides $W = 15$KHz of bandwidth for a time duration $\tau = 1$ms. Moreover, we set the number of quantization bits $b_n = 6$ bits. We consider the time-varying fading channel between the nodes to follow a Rayleigh distribution with zero mean and unit variance, i.e., $h \sim \mathcal{CN}(0, 1)$. We assume the channel coherence time is 10 iterations. Regarding the communication scheme, the number of time slots needed to transmit data from every node depends on the number of bits to be sent, as well as on the condition of the channel between the communicating nodes. In particular, we consider the required number of uploading time slots to exchange the updates from node n to node m as the minimum τ_n such that the following inequality is satisfied

$$\sum_{t=0}^{\tau_n} \sum_{s=1}^{\frac{N_s}{N}} \tau R^s_{n,m}(t) \geq D_n, \tag{31}$$

where $R^s_{n,m}(t) = W \log_2 \left(1 + P_n |h^s_{n,m}(t)|^2 / N_0 W\right)$ is the rate expression, and $N_0 = 10^{-9}$ W/Hz is the noise power spectral density. D_n is the number of transmitted bits for node n, which is algorithm-specific (for example $2 \times 32d$ for *DNGAM* and $2 \times (b_n d + (b_n + R_n))$ for *Q-DNGAM*). Consequently, we can quantify the transmission energy at node n as, $E_n = \tau_n \cdot P_n$. Throughout the simulations, we assume that we have a perfect channel estimation for the duration $t \in [0, \tau_n)$.

We compare our proposed algorithms with the baselines, and we select the simulation parameters that result in the best performance.

- **GADMM**: First-order decentralized algorithm based on ADMM where the nodes exchange the primal variables only with two neighbors ($\alpha = 1, \rho = 10, D_n = 32d$).
- **Q-GADMM**: The quantized version of *GADMM*, ($\alpha = 1, \rho = 10, D_n = b_n d + (b_n + R_n)$).
- **Newton tracking**: Second-order decentralized algorithm, where each node updates its local variable along a modified local Newton direction using neighboring and historical information, ($\epsilon = 5, \alpha = 0.07, D_n = 32d$).
- **DSGD**: First-order decentralized SGD algorithm where the nodes share their updated gradients to their neighbors, followed by averaging the model, ($\alpha = 10^{-4}, D_n = 64d$).

B. Simulation Results and Discussion

We evaluate the performance of our proposed algorithms *DNGAM* and *Q-DNGAM* (detailed in section II) in terms of the loss $|f(\boldsymbol{x}^r) - f(\boldsymbol{x}^*)|$ with respect to different metrics, where the value $f(\boldsymbol{x}^*)$ is obtained at the convergence point when performing the standard Newton's method. Fig. 1(a-c) shows the loss with respect to the number of communication rounds, the communication bits per node, and the transmission energy per node, respectively. Figs. 2 and 3 present the loss versus different signal-to-noise ratio (SNR) values for two different numbers of available channel uses (CUs). The number of CUs is defined as $CUs = \sum_{j=1}^{J} S_j$, where S_j is the number of available subcarriers at time slot j.

In Fig. 1a, we notice that both *DNGAM* and *Q-DNGAM* achieve the target loss 10^{-4} at almost the same number of communication rounds, which is low compared to first-order counterparts *(Q-)GADMM* and *DSGD*. For instance, our algorithms take at most 1023 rounds compared to 2306 for *(Q-)GADMM*. This is justified by the effect of second-order information on the convergence speed. *Newton tracking* baseline suffers from a slow convergence, which can be related to the poor estimation of the Newton direction, when only the information from two neighbors is used to minimize the consensus errors. In Fig. 1b, we show that our quantized algorithm *Q-DNGAM* transmits less number of bits per node to reach the target loss, compared to other baselines. This is due to the low number of bits per iteration that quantization

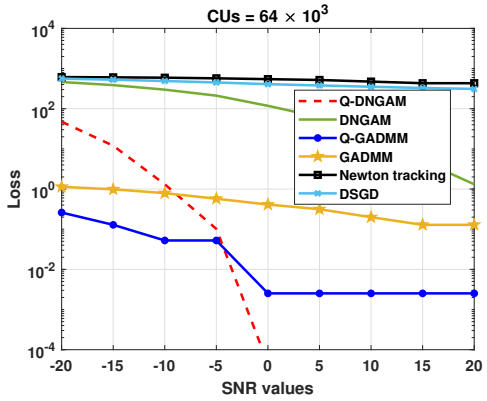

Fig. 2: Loss with respect to different SNR values for $CUs = 64 \times 10^3$.

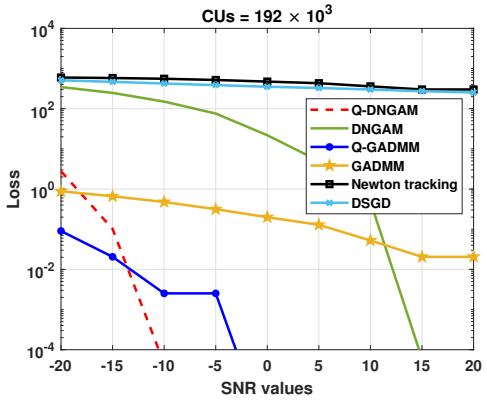

Fig. 3: Loss with respect to different SNR values for $CUs = 192 \times 10^3$.

achieves. To highlight the transmission energy aspect, Fig. 1c demonstrates the advantage of *Q-DNGAM* as an energy-efficient solution with respect to other baselines. Specifically, the first-order quantized baseline *Q-GADMM* requires more than 2.2 times transmission energy per node compared to *Q-DNGAM*.

In Figs. 2 and 3, we consider different SNR values by fixing N_0W and varying the transmission power. For Fig. 2, we limit the number of available CUs to $CUs = 64 \times 10^3$. We see that *Q-DNGAM* achieves the target loss at SNR $= 8$ dB, whereas the other algorithms fail to attain the target even for SNR$=$ 20dB. This is since *Q-DNGAM* enjoys a low number of transmitted bits per iteration and a faster convergence. We notice also that increasing the SNR value improves the performance of the baselines, except for *DSGD* and *Newton tracking*, which require high number of iterations to converge, hence more communication resources. As for Fig. 3, when we increase the number of available CUs to $CUs = 192 \times 10^3$, we observe an improvement in the performance of the algorithms. Notably, both *Q-DNGAM* and *Q-GADMM* reach the target loss starting from SNR$= -4dB$ and SNR$= -1dB$, respectively. Moreover, we notice that the quantized algorithms (*Q-DNGAM* and *Q-GADMM*) outperform their non-quantized counterparts (*DNGAM* and *GADMM*) for all SNR values, which justifies the adoption of quantization to achieve extra communication efficiency.

IV. Conclusion

In this paper, we presented a communication-efficient second-order Newton-type approach to solve the problem of decentralized FL. In particular, we reformulate the Newton step as a solution to a convex quadratic problem and approximate it using one *GADMM* step. Consequently, every node transmits two model-sized vector updates to its two neighbors, which saves communication resources. Moreover, we apply stochastic quantization for further reduction in communication costs. Simulation results illustrate that our algorithm *Q-DNGAM* outperforms the baselines and provides an energy and communication-efficient solution. Additionally, our algorithm shows a robust and enhanced performance for bandwidth-limited systems under different SNR regimes. For future directions, we aim to extend this work for generalized network topology and investigate channel estimation and synchronization between the nodes.

References

[1] M. Krouka, A. Elgabli, C. Ben Issaid, and M. Bennis, "Communication-efficient and federated multi-agent reinforcement learning," *IEEE Transactions on Cognitive Communications and Networking*, vol. 8, no. 1, pp. 311–320, 2022.

[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 *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 1–6.

[3] A. Elgabli, H. Khan, M. Krouka, and M. Bennis, "Reinforcement learning based scheduling algorithm for optimizing age of information in ultra reliable low latency networks," in *2019 IEEE Symposium on Computers and Communications (ISCC)*, 2019, pp. 1–6.

[4] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar, "signSGD: Compressed optimisation for non-convex problems," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80. PMLR, 10–15 Jul 2018, pp. 560–569.

[5] J. Wangni, J. Wang, J. Liu, and T. Zhang, "Gradient sparsification for communication-efficient distributed optimization," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS'18, 2018, p. 1306–1316.

[6] 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, 2018.

[7] 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, 2018.

[8] Y. Malitsky and K. Mishchenko, "Adaptive gradient descent without descent," in *Proceedings of the 37th International Conference on Machine Learning*, ser. ICML'20. JMLR.org, 2020.

[9] C. Xie, O. Koyejo, I. Gupta, and H. Lin, "Local adaalter: Communication-efficient stochastic gradient descent with adaptive learning rates," *arXiv preprint arXiv:1911.09030*, 2019.

[10] H. Yu, R. Jin, and S. Yang, "On the linear speedup analysis of communication efficient momentum SGD for distributed non-convex optimization," in *International Conference on Machine Learning*. PMLR, 2019, pp. 7184–7193.

[11] S. Soori, K. Mischenko, A. Mokhtari, M. M. Dehnavi, and M. Gurbuzbalaban, "DAve-QN: A distributed averaged quasi-newton method with local superlinear convergence rate," in *AISTATS*, 2020.

[12] M. Safaryan, R. Islamov, X. Qian, and P. Richtárik, "FedNL: Making newton-type methods applicable to federated learning," *International Workshop on Federated Learning for User Privacy and Data Confidentiality in Conjunction with ICML 2021 (FL-ICML'21)*, 2021.

[13] A. Koloskova, N. Loizou, S. Boreiri, M. Jaggi, and S. Stich, "A unified theory of decentralized SGD with changing topology and local updates," in *Proceedings of the 37th International Conference on Machine Learning*, vol. 119. PMLR, 13–18 Jul 2020, pp. 5381–5393.

[14] A. Elgabli, J. Park, A. S. Bedi, M. Bennis, and V. Aggarwal, "GADMM: Fast and communication efficient framework for distributed machine learning," *Journal of Machine Learning Research*, vol. 21, no. 76, pp. 1–39, 2020.

[15] A. Elgabli, J. Park, A. S. Bedi, M. Bennis, and V. Aggarwal, "Q-GADMM: Quantized group ADMM for communication efficient decentralized machine learning," in *ICASSP*, 2020, pp. 8876–8880.

[16] J. Zhang, Q. Ling, and A. M.-C. So, "A newton tracking algorithm with exact linear convergence for decentralized consensus optimization," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 7, pp. 346–358, 2021.

[17] J. Zhang, K. You, and T. Başar, "Distributed adaptive newton methods with global superlinear convergence," *Automatica*, vol. 138, p. 110156, 2022.

[18] A. Mokhtari, W. Shi, Q. Ling, and A. Ribeiro, "A decentralized second-order method with exact linear convergence rate for consensus optimization," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 4, pp. 507–522, 2016.

[19] M. Krouka, A. Elgabli, C. B. Issaid, and M. Bennis, "Communication-efficient federated learning: A second order newton-type method with analog over-the-air aggregation," *IEEE Transactions on Green Communications and Networking*, vol. 6, no. 3, pp. 1862–1874, 2022.

[20] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, may 2011.