

# DIN: A Decentralized Inexact Newton Algorithm for Consensus Optimization

Abdulmomen Ghalkha, Chaouki Ben Issaid, Anis Elgabli and Mehdi Bennis

Centre for Wireless Communications (CWC)  
University of Oulu, Finland

Email: {abdulmomen.ghalkha, chaouki.benissaid, anis.elgabli, mehdi.bennis}@oulu.fi

**Abstract**—In this paper, we consider a decentralized consensus optimization problem defined over a network of inter-connected devices that collaboratively solve the problem using only local data and information exchange with their neighbours. Despite their fast convergence, Newton-type methods require sending Hessian information between devices, making them communication inefficient while violating the devices’ privacy. By formulating the Newton direction learning problem as a sum of separable functions subjected to a consensus constraint, our proposed approach learns an inexact Newton direction alongside the global model using the proximal primal-dual (Prox-PDA) algorithm. Our algorithm, coined DIN, avoids sharing Hessian information between devices since each device shares a model-sized vector, concealing the first- and second-order information, reducing the network’s burden and improving communication and energy efficiencies. Numerical simulations corroborate that DIN exhibits higher communication efficiency in terms of communication rounds while consuming less communication and computation energy compared to existing second-order decentralized baselines.

**Index Terms**—Distributed optimization, decentralized learning, communication-efficient federated learning, second-order methods.

## I. INTRODUCTION

Minimizing the sum of functions in a distributed manner is motivated by wide applications in various networked systems, such as smart grids [1], federated learning (FL) [2], and wireless sensor networks [3]. A traditional approach is to use a central parameter server (PS), which has high computational and storage capabilities. Each device sends its raw data to the PS, which applies centralized optimization to minimize the global objective function. Although this traditional approach is simple, it suffers from high communication costs and violates privacy. To enable collaborative learning while protecting privacy, privacy-preserving collaborative learning techniques are necessary. Recently, thanks to the fast growth of the computation power of edge clients, using FL transmission of raw and private data to the PS can be avoided. In the canonical FL approach, local models/gradients are updated locally, and the PS aggregates the local models/gradients to update the global model/gradient, which is then shared with edge clients. Iterating this way, eventually, all clients converge to a global model.

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

Existing FL algorithms can be categorized into three groups based on which information from the objective function is used in the optimization process. Zeroth-order algorithms are the first category in which clients are limited to using samples of their own objective functions [4]. The second category is first-order algorithms where edge clients use the gradients of their objective functions, which decide the direction of the update. Primal methods such as federated averaging [5], and primal-dual methods such as distributed alternating direction method of multipliers (ADMM) [6] are examples of first-order algorithms. Second-order algorithms, which are the last category, employ the objective function’s second-order information, i.e., the Hessian matrix, at each iteration. Despite the fast convergence of Newton’s method, which is the standard second-order algorithm, it suffers from high communication costs. Moreover, it introduces privacy issues, since the Hessian matrix contains important information about the characteristics of the local objective function and data. For instance, the authors in [7] demonstrated how information from input images can be extracted using the eigenvalues of the Hessian matrix. The aforementioned frameworks require a PS to aggregate the first- and second-order data received from edge clients. Relying on a single PS may introduce a lot of communication overhead and in a large system may not even be possible. Moreover, as an aggregation hub, the network may experience a single point of failure. Therefore fully decentralized approaches, where there is no central PS, have been gaining popularity. In fully decentralized FL, edge clients share their local information with their neighboring clients to establish model consensus, which avoids creating a single point of failure while reducing the communication bottleneck that occurs at the PS [8]–[10].

### A. Related Works

Communication-efficient solutions for distributed optimization have been a study subject of several articles. The following discussion highlights various techniques.

1) *First-Order Methods*: The standard approach to solving the distributed optimization problem in the PS-based topology is to use first-order methods such as distributed gradient descent (DGD). At every iteration of DGD, each client computes its local gradient with respect to the current model parameters and sends that information to the PS. After receiving all

gradients, the PS computes the global gradient and executes one GD step. In decentralized settings, the local gradients are shared among neighboring clients where each client averages the received gradients and then performs a local GD step to update its local model. Although first-order methods enjoy low computation complexity, they suffer from a slow convergence rate, which depends on the condition number. This calls for a large number of communication rounds; in addition to considerable energy and bandwidth resources per communication round. This bottleneck can be tackled by either reducing the number of communication rounds to ensure fast convergence or minimizing the communication overhead per communication round using compression schemes. Several techniques were proposed to reduce the number of communication rounds by accelerating the convergence using momentum [11], and adaptive learning rate [12]. On the other hand, imposing quantization [13] and censoring [14] can help reduce the communication cost per iteration.

2) *Second-Order Methods*: Recently, second-order algorithms have attracted a lot of attention because they achieve faster convergence compared to first-order techniques taking advantage of the second derivative’s curvature information, which gives adaptive update directions. Although this reduces the number of communication rounds, second-order information necessitates significant computation and communication costs. In every communication round, the Hessian matrix is computed and transmitted, which creates a communication cost of  $\mathcal{O}(n^2)$  per iteration compared to  $\mathcal{O}(n)$  in first-order methods, where  $n$  is the dimension of the model. Furthermore, Newton’s approach is sensitive to inversion attacks since it involves sharing both the gradient and the Hessian at each iteration, which is a major concern in distributed systems [15].

The problem of sending the exact Hessian matrix has been addressed in various studies with communication-efficient solutions that avoid sending the exact Hessian. The authors in [16] suggested a Newton-based framework, in which edge clients communicate a compressed version of the local Hessian. However, gradients and compressed Hessians are still communicated; hence the privacy issue is not completely addressed. In a recent work [17], the privacy issue was solved by learning the inverse Hessian-gradient product. The idea is to formulate an inner problem with the objective of learning the inverse Hessian-gradient. One ADMM step is performed at the clients in every iteration to approximate the solution of the inner problem, then the output is shared with the PS. This algorithm still requires the existence of the PS to aggregate the received directions and form the global Newton direction. In this work, we extend this idea to the fully decentralized setting to solve the aforementioned drawbacks of PS-based solutions. Few works have utilized second-order information in decentralized settings to accelerate convergence. In [18], authors approximate the exact Newton step by truncating  $K$  terms of the Taylor series expansion. However, this algorithm requires multiple exchanges of the local directions to approximate the exact Hessian, which calls for more communication rounds. Authors in [19] incorporate the local Hessian in the

update direction while tracking the gradient. However, the local Newton direction may not be a good estimate for the global one. Throughout this paper, we will refer to both algorithms as Network Newton (NN) and Newton Tracking (NT), respectively, and refer to them as baselines.

## B. Contributions and Outline

In this paper, we propose DIN, a second-order based, decentralized, and communication-efficient FL scheme that reduces the communication overhead per iteration and preserves privacy by concealing the gradient and the Hessian. DIN learns the inverse Hessian-gradient product alongside the model. The problem of learning the inverse Hessian-gradient product is formulated as a constrained optimization problem and a framework based on Prox-PDA is used to learn  $\nabla^2 f(\mathbf{x})^{-1} \nabla f(\mathbf{x})$ . In contrast to  $\mathcal{O}(n^2)$  in standard Newton, each client in this step shares a model-sized vector, yielding  $\mathcal{O}(n)$  communication complexity per iteration. Each client updates its model utilizing the inexact Newton step using the average estimates of the newton direction received from the neighboring clients. Our contribution can be summarized as follows

- We propose DIN algorithm, a decentralized framework that uses second-order information to solve the consensus optimization problem. More specifically, we use Prox-PDA [10] algorithm to tackle the problem of learning the inverse-Hessian-gradient product by decomposing the global inverse-Hessian-gradient product learning function into a sum of separable local functions.
- Our proposed algorithm is communication and energy-efficient and privacy-preserving at each iteration. DIN does not require clients to share their explicit gradient and Hessian matrix at any iteration, resulting in a communication cost of  $\mathcal{O}(n)$  and privacy preservation.
- We conducted a number of experiments to solve the decentralized logistic regression problem with real datasets while monitoring energy consumption. Numerical results show that DIN outperforms Network Newton and Newton Tracking methods under different network topologies and graph densities. We show that DIN consumes less energy to achieve the same optimality gap.

The paper is structured as follows. In Section II, we describe the system model and problem formulation. In Section III, we describe our proposed algorithm. Then, we conduct several numerical experiments to compare the performance of DIN with key baselines in Section IV. Finally, we give a conclusion of our work in Section V.

## II. SYSTEM MODEL AND PROBLEM FORMULATION

We consider a connected network consisting of  $N$  devices, each having a local loss function  $f_i : \mathbb{R}^d \rightarrow \mathbb{R}$ , assumed to be a convex and second-order differentiable, known only to device  $i$ . The devices are connected through a graph  $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ , where  $\mathcal{V}$  and  $\mathcal{E}$  are the node and edge sets, respectively. Devices collaborate to minimize the empirical loss/risk, i.e., the average of their local objective functions, to find a common

model,  $\mathbf{x} \in \mathbb{R}^d$ . Every device  $i$  can only communicate with its immediate neighbors, defined as  $\mathcal{N}_i = \{j | (i, j) \in \mathcal{E}\}$ , with  $|\mathcal{N}_i| = \delta_i$  denote the cardinality of its neighbor set. Specifically, the devices' goal is to find the model that solves the following learning problem in a decentralized manner

$$(P1) \quad \min_{\mathbf{x} \in \mathbb{R}^d} f(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^N f_i(\mathbf{x}). \quad (1)$$

The starting point of our work is the Newton-like method introduced in [20], which solves (P1) in the presence of a PS. At iteration  $(k+1)$ , the *Newton* step update is given by

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \left( \frac{1}{N} \sum_{i=1}^N \nabla^2 f_i(\mathbf{x}^k) \right)^{-1} \left( \frac{1}{N} \sum_{i=1}^N \nabla f_i(\mathbf{x}^k) \right), \quad (2)$$

where  $\nabla^2 f_i(\cdot) \in \mathbb{R}^{d \times d}$  and  $\nabla f_i(\cdot)$  are the Hessian and the gradient of  $f_i(\cdot)$ , respectively. For ease of notations, we define  $\mathbf{H}_i^k = \nabla^2 f_i(\mathbf{x}^k)$  and  $\mathbf{g}_i^k = \nabla f_i(\mathbf{x}^k)$  as the Hessian matrix and the gradient vector of device  $i$  evaluated at  $\mathbf{x}^k$ , respectively. We also define the network Hessian and gradient as

$$\bar{\mathbf{H}}^k = \frac{1}{N} \sum_{i=1}^N \mathbf{H}_i^k \quad \text{and} \quad \bar{\mathbf{g}}^k = \frac{1}{N} \sum_{i=1}^N \mathbf{g}_i^k. \quad (3)$$

Hence, we can write the step in (2) as follows

$$\mathbf{x}^{k+1} = \mathbf{x}^k - (\bar{\mathbf{H}}^k)^{-1} \bar{\mathbf{g}}^k. \quad (4)$$

Note that (4) can be implemented if every device has access to the average of all gradients and all Hessians evaluated at  $\mathbf{x}^k$ . However, this update cannot be implemented in a decentralized way since every device can only exchange information with a limited number of neighbors; thus it cannot obtain  $\bar{\mathbf{g}}^k$  and  $\bar{\mathbf{H}}^k$ . Before we present our algorithm, we start by introducing matrices related to the network topology

- The degree matrix  $\tilde{\mathbf{D}} = \text{diag}[\delta_1, \delta_2, \dots, \delta_N]$ , a diagonal matrix containing the number of neighbors of each device, i.e., the degree of the device  $i$ .
- The incidence matrix  $\tilde{\mathbf{A}}$  with entries  $\tilde{\mathbf{A}}(k, i) = 1$  and  $\tilde{\mathbf{A}}(k, j) = -1$  if  $k = (i, j) \in \mathcal{E}$  with  $j > i$ .
- The signed and signless Laplacian matrices defined as  $\tilde{\mathbf{L}}_- = \tilde{\mathbf{A}}^T \tilde{\mathbf{A}}$  and  $\tilde{\mathbf{L}}_+ = 2\tilde{\mathbf{D}} - \tilde{\mathbf{L}}_-$ , respectively.

We also define the extended versions of these matrices where the extended definition is given by taking the Kronecker product with the identity matrix, i.e.,  $\mathbf{A} = \tilde{\mathbf{A}} \otimes \mathbf{I}_d$ .

### III. PROPOSED ALGORITHM

Inspired by [17], we propose to replace the *inverse Hessian-gradient product*, i.e., the term  $(\bar{\mathbf{H}}^k)^{-1} \bar{\mathbf{g}}^k$  in (4), with an approximate solution of the following optimization problem

$$\mathbf{d}^k = \arg \min_{\mathbf{d} \in \mathbb{R}^d} \frac{1}{2} \mathbf{d}^T \bar{\mathbf{H}}^k \mathbf{d} - \mathbf{d}^T \bar{\mathbf{g}}^k. \quad (5)$$

Specifically, when solving the problem in (5) at iteration  $k$ , we find the direction  $\mathbf{d}^k = (\bar{\mathbf{H}}^k)^{-1} \bar{\mathbf{g}}^k$ . Nevertheless, the solution to this problem in a decentralized manner is still not possible.

To this end, we reformulate the problem in (5) and cast it as a decentralized optimization problem

$$(P2) \quad (\mathbf{d}^*)^k = \arg \min_{\{\mathbf{d}_i\}_{i=1}^N \in \mathbb{R}^d} \left\{ \phi^k(\mathbf{d}) = \sum_{i=1}^N \phi_i^k(\mathbf{d}_i) \right\} \\ \text{s.t.} \quad \mathbf{d}_i = \mathbf{d}_j, \quad \forall (i, j) \in \mathcal{E}, \quad (6)$$

where  $\phi_i^k(\mathbf{d}_i) = \frac{1}{2} \mathbf{d}_i^T (\mathbf{H}_i^k + \alpha \mathbf{I}_d) \mathbf{d}_i - \mathbf{d}_i^T \mathbf{g}_i^k$ ,  $\alpha$  is a hyperparameter that we introduce to make sure that the matrix  $(\mathbf{H}_i^k + \alpha \mathbf{I}_d)$  invertible, and  $\mathbf{d} = [\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_N]^T \in \mathbb{R}^{Nd}$  the concatenation of the local directions. Note that the inexact Newton direction, i.e.,  $-(\mathbf{H}_i^k + \alpha \mathbf{I}_d)^{-1} \mathbf{g}_i^k$ , is also a valid descent direction [19]. For a given  $\mathbf{x}_i^k$ , solving (P2) exactly, i.e., until converging to  $(\mathbf{d}^*)^k$ , comes at a very high communication cost since devices need to iterate and communicate their updates at each iteration until convergence. In this work, we propose to perform a single update at each outer iteration  $k$  to approximate the solution of (P2) and reduce the communication cost. In what follows, we elaborate on how the single pass update of the direction  $\mathbf{d}$  is done. Using these introduced notations, (P2) can be re-written as

$$\min_{\mathbf{d} \in \mathbb{R}^{Nd}} \phi^k(\mathbf{d}) \\ \text{s.t.} \quad \mathbf{A} \mathbf{d} = \mathbf{0} \quad (7)$$

The augmented Lagrangian of (7) is given as

$$\mathcal{L}_\rho^k(\mathbf{d}, \boldsymbol{\mu}) = \phi^k(\mathbf{d}) + \langle \boldsymbol{\mu}, \mathbf{A} \mathbf{d} \rangle + \frac{\rho}{2} \|\mathbf{A} \mathbf{d}\|^2, \quad (8)$$

where  $\rho > 0$  is a constant penalty parameter, and  $\boldsymbol{\mu} = [\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_N]^T \in \mathbb{R}^{Nd}$  is the concatenation of the dual variables. Minimizing the augmented Lagrangian directly leads to a solution that cannot be implemented in a decentralized way. Instead, we leverage the Prox-PDA algorithm [10], which adds the proximal term  $\frac{\rho}{2} \|\mathbf{d} - \mathbf{d}^{k-1}\|_{\tilde{\mathbf{L}}_+}^2$ . In this case, the update of the primal variables, at iteration  $k$ , is given

---

#### Algorithm 1 Decentralized Inexact Newton (DIN)

---

- 1: **Input:**  $N, \{f_i(\cdot)\}_{i=1}^N, \rho, K$ ,
- 2: **Output:**  $\mathbf{x}, \forall i$
- 3: **Initialization:**  $\mathbf{x}_i^0, \mathbf{d}_i^{(-1)}, \boldsymbol{\lambda}_i^{(-1)}, \forall i$ .
- 4: **for**  $k = 0, \dots, K$  **do**
- 5:   **Every node in parallel**
- 6:   Computes its Newton direction using

$$\mathbf{d}_i^k = (\mathbf{H}_{i,\alpha}^k)^{-1} \left( \mathbf{g}_i^k - \boldsymbol{\lambda}_i^{k-1} + \rho \left( \delta_i \mathbf{d}_i^{k-1} + \sum_{j \in \mathcal{N}_i} \mathbf{d}_j^{k-1} \right) \right).$$

- 7:   Updates its dual variable via

$$\boldsymbol{\lambda}_i^k = \boldsymbol{\lambda}_i^{k-1} + \rho \left( \delta_i \mathbf{d}_i^k - \sum_{j \in \mathcal{N}_i} \mathbf{d}_j^k \right).$$

- 8:   Updates its local model using  $\mathbf{x}_i^{k+1} = \mathbf{x}_i^k - \mathbf{d}_i^k$ .
  - 9: **end for**
-

by solving the following optimization problem [10]

$$\min_{\mathbf{d} \in \mathbb{R}^{Nd}} \phi^k(\mathbf{d}) + \langle \boldsymbol{\mu}^{k-1}, \mathbf{A}\mathbf{d} \rangle + \frac{\rho}{2} \|\mathbf{A}\mathbf{d}\|^2 + \frac{\rho}{2} \|\mathbf{d} - \mathbf{d}^{k-1}\|_{\mathbf{L}_+}^2. \quad (9)$$

Using  $\mathbf{L}_- = \mathbf{A}^T \mathbf{A}$  and  $2\mathbf{D} = \mathbf{L}_- + \mathbf{L}_+$ , we can write

$$\min_{\mathbf{d} \in \mathbb{R}^{Nd}} \phi^k(\mathbf{d}) + \langle \boldsymbol{\mu}^{k-1}, \mathbf{A}\mathbf{d} \rangle + \rho \mathbf{d}^T \mathbf{D} \mathbf{d} - \rho \mathbf{d}^T \mathbf{L}_+ \mathbf{d}^{k-1}. \quad (10)$$

Setting the derivative with respect to  $\mathbf{d}$  to zero, we get

$$\nabla \phi^k(\mathbf{d}^k) + \mathbf{A}^T \boldsymbol{\mu}^{k-1} + 2\rho \mathbf{D} \mathbf{d}^k - \rho \mathbf{L}_+ \mathbf{d}^{k-1} = \mathbf{0}. \quad (11)$$

On the other hand, the update of  $\boldsymbol{\mu}$ , at iteration  $k$ , is given by

$$\boldsymbol{\mu}^k = \boldsymbol{\mu}^{k-1} + \rho \mathbf{A} \mathbf{d}^k. \quad (12)$$

Next, we define  $\boldsymbol{\lambda} = \mathbf{A}^T \boldsymbol{\mu}$  and multiply both sides in (12) by  $\mathbf{A}^T$ . Using the fact that  $\mathbf{L}_- = \mathbf{A}^T \mathbf{A}$ , we get

$$\boldsymbol{\lambda}^k = \boldsymbol{\lambda}^{k-1} + \rho \mathbf{L}_- \mathbf{d}^k. \quad (13)$$

Hence, the dual variable of the  $i^{\text{th}}$  device is updated as

$$\boldsymbol{\lambda}_i^k = \boldsymbol{\lambda}_i^{k-1} + \rho \left( \delta_i \mathbf{d}_i^k - \sum_{j \in \mathcal{N}_i} \mathbf{d}_j^k \right). \quad (14)$$

Writing the update of the primal variable of the  $i^{\text{th}}$  device from (7), we get

$$\nabla \phi_i^k(\mathbf{d}_i^k) + \boldsymbol{\lambda}_i^{k-1} + 2\rho \delta_i \mathbf{d}_i^k - \rho \left( \delta_i \mathbf{d}_i^{k-1} + \sum_{j \in \mathcal{N}_i} \mathbf{d}_j^{k-1} \right) = \mathbf{0}. \quad (15)$$

Replacing the expression of  $\nabla \phi^k(\mathbf{d}^k)$  and re-arranging the terms, we can write

$$\mathbf{d}_i^k = (\mathbf{H}_{i,\alpha}^k)^{-1} \left( \mathbf{g}_i^k - \boldsymbol{\lambda}_i^{k-1} + \rho \left( \delta_i \mathbf{d}_i^{k-1} + \sum_{j \in \mathcal{N}_i} \mathbf{d}_j^{k-1} \right) \right), \quad (16)$$

where  $\mathbf{H}_{i,\alpha}^k = \mathbf{H}_i^k + (2\rho \delta_i + \alpha) \mathbf{I}_d$ . Finally, the local model is updated using the local Newton direction as

$$\mathbf{x}_i^{k+1} = \mathbf{x}_i^k - \mathbf{d}_i^k. \quad (17)$$

The details of our algorithm are summarized in Algorithm 1.

#### IV. NUMERICAL EVALUATION

In this section, we conduct numerical experiments to evaluate the performance of our proposed algorithm DIN, against first- and second-order algorithms, DGD, Network Newton (NN) [18], and Newton Tracking (NT) [19], under different network topologies. We consider a binary classification problem using regularized logistic regression.

##### A. Experimental Setup

We consider the regularized logistic regression problem

$$\min_{\mathbf{x} \in \mathbb{R}^d} \{f(\mathbf{x}) := \frac{1}{N} \sum_{i=1}^N f_i(\mathbf{x}) + \frac{\eta}{2} \|\mathbf{x}\|^2\}, \quad (18)$$

where the local loss function  $f_i(\mathbf{x})$  is defined as

$$f_i(\mathbf{x}) = \frac{1}{m} \sum_{j=1}^m \log(1 + \exp(-b_{ij} \mathbf{a}_{ij}^T \mathbf{x})), \quad (19)$$

$\{a_{ij}, b_{ij}\}_{j=1, \dots, m}$  denote the data points at the  $i^{\text{th}}$  device ( $i \in \{1, \dots, N\}$ ), where  $m$  represents the number of data samples of each device. A regularization parameter  $\eta > 0$  is added to avoid overfitting and chosen to be equal to  $10^{-3}$ . We consider three real datasets: a9a, w8a, and phishing, that were taken from LibSVM [21]. The data is evenly split between  $N$  workers, which are connected with undirected edges of a given generated graph. The number of features of each dataset and the number of workers is depicted in Table I. Four network topologies are implemented in the experiments: a binomial graph with edge creation probability  $p = 0.4$ , a geometric graph with distance  $d = 0.4$ , as well as the grid and ring topologies. The energy footprint of the  $i^{\text{th}}$  device consumed during training consists of two parts, computation and communication components. The computation component  $E_c$  consists of the energy required to power the hardware (e.g., CPUs, GPUs, Memories, etc.), while the latter  $E_t$  represents the energy needed to transmit and receive bits between neighboring devices [22]. The total energy consumed by device  $i$  after  $t$  iterations can be written as

$$E_T(t) = E_c(t) + E_t(t), \quad (20)$$

with

$$E_c(t) = \sum_{k=1}^t e_{\text{device},i} \quad \text{and} \quad E_t(t) = \sum_{k=1}^t \sum_{j \in \mathcal{N}_i} b(\mathbf{d}_i^k) e_{i,j}, \quad (21)$$

where  $e_{\text{device},i}$  is the computation energy consumed by device  $i$  to perform one iteration,  $b(\mathbf{d}_i^k)$  is the size of the inverse Hessian-gradient product vector in bits, and  $e_{i,j}$  is the energy needed to transmit one bit from device  $i$  to neighbour  $j$ . We conduct the experiment on an NVIDIA Jetson Dev Board and we monitor the energy efficiency and the carbon emission using eco2AI python library [23]. The devices are randomly distributed over a  $100 \times 100 \text{ m}^2$  area, and we assume a digital communication link with a free-space path loss channel model. Hence, the maximum achievable rate  $R = B \log_2(1 + \frac{P_t}{d_{i,j}^2 B N_0})$ , where  $B$  is the bandwidth,  $P_t$  is the transmission power,  $d_{i,j}$  is the distance between transmitter  $i$  and receiver  $j$ , and  $N_0$  is the noise spectral density. To find the maximum data rate between neighbouring devices and the energy consumed for transmission, we assume each device transmits at full power  $P_t = 100 \text{ mW}$ ,  $B = 2 \text{ MHz}$ ,  $N_0 = 10^{-9} \text{ W/Hz}$ , and a 32-bit representation of transmitted elements.

TABLE I: Details of the datasets

Dataset	$n$	$m$	$d$	$N$
a9a	32560	407	109	80
w8a	49700	350	267	142
phishing	11000	110	68	100

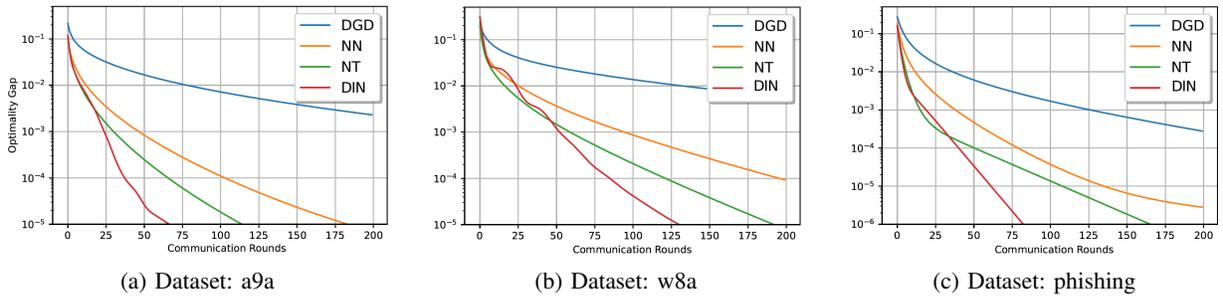


Fig. 1: Optimality gap of DIN compared to baselines in terms of the number of communication rounds for different datasets.

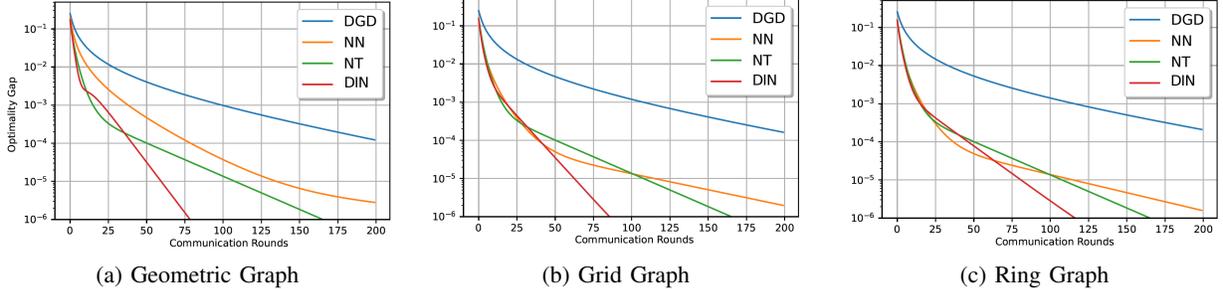


Fig. 2: Optimality gap versus the number of communication rounds for (a) geometric, (b) grid, and (c) ring network topologies using the phishing dataset.

To evaluate the performance of the aforementioned algorithms, we plot the optimality gap  $f(\bar{\mathbf{x}}^k) - f(\mathbf{x}^*)$  as a function of the number of communication rounds, where  $\mathbf{x}^*$  and  $f(\mathbf{x}^*)$  are pre-computed using standard Newton’s method until convergence and  $\bar{\mathbf{x}}^k$  is the average model at iteration  $k$ . For hyperparameters tuning, we pick the parameters that lead to the best performance for each algorithm in all experiments.

### B. Performance Comparison

Fig. 1 illustrates the optimality gap as a function of the number of communication rounds in a decentralized network topology with a connection probability  $p = 0.4$ . We observe from Fig. 1 that DIN is the fastest, followed by Newton Tracking, Network Newton, and DGD for the three datasets. We clearly see from Fig. 1-(a-c) that DIN reaches the optimality gap of  $10^{-5}$  within at least 50 communication rounds earlier than the fastest baseline (Newton Tracking). Since each algorithm has the same communication overhead per round, DIN is the most communication/energy efficient one; thanks to the fast convergence in terms of communication rounds. In Fig. 2, we investigate the performance of DIN with different network topologies. Each sub-figure plots the optimality gap with respect to the number of communication rounds for a different topology. In Fig. 2-(a), the graph is geometric where every two devices are connected if they’re located within a normalized distance  $d = 0.4$ . For the grid topology (Fig. 2-(b)), each device has a communication link with at most four neighbors. In Fig. 2-(c), the graph topology forms a ring, and each device communicates with only two neighbors. The dataset used for this comparison is the phishing dataset. We observe from Fig. 2(a-c) that DIN converges faster than the

considered baselines, although there is a degradation in the convergence speed in the case of ring topology due to the high sparsity of the network.

### C. Energy-Efficiency and Carbon footprint

In table II, we report the energy consumption and the carbon footprint required by the four algorithms to achieve a  $10^{-5}$  optimality gap using the phishing dataset. We observe that DGD’s total energy consumption is the highest, and so is its carbon footprint. Although DGD is computationally less expensive, it requires a very large number of communication rounds to achieve the target optimality gap inducing high communication energy cost. On the other hand, Network Newton consumes the highest computation energy since Network Newton performs two matrix inversion operations in each communication round. Finally, DIN requires lower energy for both computation and communication due to its fast convergence while performing a single matrix inversion operation in each communication round.

### D. Impact of the Graph Density

Now, we investigate the effect of the graph density on the convergence speed of DIN. We use four topologies: the line graph, random graphs with  $p \in \{0.3, 0.5\}$ , and the complete graph. The hyperparameters  $\alpha$  and  $\rho$  are tuned to give the fastest convergence. Fig. 3 shows the optimality gap versus the number of communication rounds. We observe that the complete graph gives the fastest speed, whereas the line graph yields the slowest convergence among all topologies. Furthermore, when  $p = 0.5$ , DIN still achieves a comparable performance to the complete graph case indicating that DIN is still applicable in networks with limited connectivity.

TABLE II: Computation/Communication energy costs and corresponding carbon footprints for the phishing dataset for a target optimality gap  $10^{-5}$ .

Algorithm	Comp. Energy [J]	Comm. Energy [J]	Total Footprint [g-CO <sub>2</sub> -eq]
DGD	$5.97E-2$ 	25.54 	$4.55E-4$
DIN	$1.32E-2$ 	4.60 	$0.82E-4$
Newton Tracking	$2.69E-2$ 	7.88 	$1.41E-4$
Network Newton	$7.96E-2$ 	10.21 	$1.83E-4$

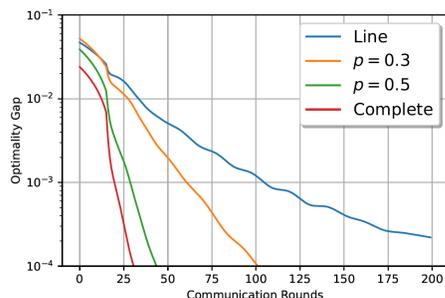


Fig. 3: Effect of the network density on the DIN performance for the a9a dataset.

## V. CONCLUSION

This paper presents a decentralized FL algorithm based on Newton’s method. Each client updates its model utilizing an approximate of the global inverse Hessian gradient product, which is calculated using its local function/data and shared approximate directions of its neighbors. By performing one Prox-PDA step, the proposed approach avoids sharing the Hessian of the device and thus ensures privacy. Furthermore, by only sharing a model-sized vector, DIN has the same per iteration communication efficiency as first-order methods, yet it is shown to be much faster and more energy-efficient. Numerical results show the supremacy of DIN over existing decentralized algorithms such as DGD, Network Newton, and Newton Tracking. The convergence analysis and the utilization of quantization for DIN are left as future work.

## REFERENCES

- [1] S. Nabavi, J. Zhang, and A. Chakraborty, “Distributed optimization algorithms for wide-area oscillation monitoring in power systems using interregional PMU-PDC architectures,” *IEEE Transactions on Smart Grid*, vol. 6, no. 5, pp. 2529–2538, 2015.
- [2] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, “Advances and open problems in federated learning,” *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.
- [3] A. Chavez, A. Moukas, and P. Maes, “Challenger: A multi-agent system for distributed resource allocation,” in *Proceedings of the first international conference on Autonomous agents*, 1997, pp. 323–331.
- [4] S. Liu, P.-Y. Chen, B. Kailkhura, G. Zhang, A. O. Hero III, and P. K. Varshney, “A primer on zeroth-order optimization in signal processing and machine learning: Principles, recent advances, and applications,” *IEEE Signal Processing Magazine*, vol. 37, no. 5, pp. 43–54, 2020.
- [5] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.
- [6] R. Zhang and J. Kwok, “Asynchronous distributed ADMM for consensus optimization,” in *International conference on machine learning*. PMLR, 2014, pp. 1701–1709.

- [7] X. Yin, B. W. 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, p. e95943, 2014.
- [8] S. Pu and A. Nedić, “Distributed stochastic gradient tracking methods,” *Mathematical Programming*, vol. 187, no. 1, pp. 409–457, 2021.
- [9] K. Yuan, Q. Ling, and W. Yin, “On the convergence of decentralized gradient descent,” *SIAM Journal on Optimization*, vol. 26, no. 3, pp. 1835–1854, 2016.
- [10] M. Hong, D. Hajinezhad, and M.-M. Zhao, “Prox-PDA: The proximal primal-dual algorithm for fast distributed nonconvex optimization and learning over networks,” in *International Conference on Machine Learning*. PMLR, 2017, pp. 1529–1538.
- [11] J. Wang, V. Tantia, N. Ballas, and M. Rabbat, “SlowMo: Improving communication-efficient distributed SGD with slow momentum,” *arXiv preprint arXiv:1910.00643*, 2019.
- [12] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečný, S. Kumar, and H. B. McMahan, “Adaptive federated optimization,” *arXiv preprint arXiv:2003.00295*, 2020.
- [13] 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 Transactions on Communications*, vol. 69, no. 1, pp. 164–181, 2020.
- [14] C. Ben Issaid, A. Elgabli, J. Park, M. Bennis, and M. Debbah, “Communication efficient decentralized learning over bipartite graphs,” *IEEE Transactions on Wireless Communications*, vol. 21, no. 6, pp. 4150–4167, 2022.
- [15] M. Fredrikson, S. Jha, and T. Ristenpart, “Model inversion attacks that exploit confidence information and basic countermeasures,” in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015, pp. 1322–1333.
- [16] M. Safaryan, R. Islamov, X. Qian, and P. Richtárik, “FedNL: Making Newton-type methods applicable to federated learning,” *arXiv preprint arXiv:2106.02969*, 2021.
- [17] A. Elgabli, C. B. Issaid, A. S. Bedi, K. Rajawat, M. Bennis, and V. Aggarwal, “FedNew: A communication-efficient and privacy-preserving Newton-type method for federated learning,” in *International Conference on Machine Learning*. PMLR, 2022, pp. 5861–5877.
- [18] A. Mokhtari, Q. Ling, and A. Ribeiro, “Network Newton distributed optimization methods,” *IEEE Transactions on Signal Processing*, vol. 65, no. 1, pp. 146–161, 2016.
- [19] 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.
- [20] R. Islamov, X. Qian, and P. Richtárik, “Distributed second order methods with fast rates and compressed communication,” in *International Conference on Machine Learning*. PMLR, 2021, pp. 4617–4628.
- [21] 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. [Online]. Available: <https://doi.org/10.1145/1961189.1961199>
- [22] S. Savazzi, V. Rampa, S. Kianoush, and M. Bennis, “An energy and carbon footprint analysis of distributed and federated learning,” *IEEE Transactions on Green Communications and Networking*, 2022.
- [23] S. Budenny, V. Lazarev, N. Zakharenko, A. Korovin, O. Plosskaya, D. Dimitrov, V. Arkhipkin, I. Oseledets, I. Barsola, I. Egorov *et al.*, “Eco2ai: carbon emissions tracking of machine learning models as the first step towards sustainable ai,” *arXiv preprint arXiv:2208.00406*, 2022.