Communication Efficient Distributed Newton Method over Unreliable Networks

Ming Wen[1], **Chengchang Liu**[2], **Yuedong Xu**[1]

[1]Fudan University,
[2]The Chinese University of HongKong
mwen23@m.fudan.edu.cn, ccliu22@cse.cuhk.hk, ydxu@fudan.edu.cn

Abstract

Distributed optimization in resource constrained devices demands both communication efficiency and fast convergence rates. Newton-type methods are getting preferable due to their superior convergence rates compared to the first-order methods. In this paper, we study a new problem in regard to the second-order distributed optimization over unreliable networks. The working devices are power-limited or operate in unfavorable wireless channels, experiencing packet losses during their uplink transmission to the server. Our scenario is very common in real-world and leads to instability of classical distributed optimization methods especially the second-order methods because of their sensitivity to the imprecision of local Hessian matrices. To achieve robustness to high packet loss, communication efficiency and fast convergence rates, we propose a novel distributed second-order method, called **RED-New** (Packet loss **Re**silient **D**istributed Approximate **New**ton). Each iteration of **RED-New** comprises two rounds of light-weight and lossy transmissions, in which the server aggregates the local information with a new developed scaling strategy. We prove the linear-quadratic convergence rate of **RED-New**. Experimental results demonstrate its advantage over first-order and second-order baselines, and its tolerance to packet loss rate ranging from 5% to 40%.

Introduction

Distributed optimization has gained prominence in machine learning due to its significant role in improving the efficiency of model training (Boyd 2011; Li et al. 2014; McMahan et al. 2017; Li et al. 2020). A typical distributed optimization (interchangeable with distributed machine learning (DML)) problem is expressed in the form

$$\min_{\mathbf{w}\in\mathbb{R}^d} f(\mathbf{w}) := \frac{1}{m}\sum_{i=1}^{m} f_i(\mathbf{w}), \tag{1}$$

where d denotes the dimension of the model $\mathbf{w}$, m is the number of distributed nodes, $f_i(\mathbf{w})$ is the local smooth loss function related to i-th client, and $f(\mathbf{w})$ is the global empirical risk to minimize.

A range of optimization algorithms are available to solve a DML problem. Among them, first-order methods and second-order methods are prevalently utilized (Konečný et al. 2017; Mitra et al. 2021; Lee et al. 2017; Li et al. 2020; Ding et al. 2022; Mishchenko et al. 2022). However, the main drawback of the first-order distributed methods is their slow convergence rate which leads to large amount of communication rounds. Compared to this, the second-order algorithms, by computing the local curvature of the objective function, always outperform first-order methods in terms of convergence rate (Nesterov and Polyak 2006; Nesterov 2007) and have been adapted to operate within a distributed framework to speed up the training process.

Implementing the Newton method within a distributed architecture encounters two primary obstacles as following:

- **Insufficient algorithm robustness:** The Newton method requires calculating the inverse of the Hessian. When dealing with an ill-conditioned Hessian matrix, even minor errors in the calculation or transmission of the Hessian matrix can lead to large deviation in the inverse, thereby impacting the convergence of the distributed algorithm.
- **Inefficient communication per-round:** In each round of the distributed Newton method, local Hessian matrix needs to be transmitted with the data volume of $O(d^2)$, which may offset the communication efficiency brought by fast convergence.

Previous literature on distributed second-order Newton methods primarily focus on addressing the second challenge (Shamir, Srebro, and Zhang 2014; Smith et al. 2018; Islamov, Qian, and Richtarik 2021; Elgabli et al. 2022; Liu et al. 2023). However, a critical aspect often overlooked is the assumption that the distributed training system operates within a reliable and stable network environment. In cross-device distributed scenarios, participants in the training can range from mobile phones to IoT (Internet of Things) nodes. These devices, susceptible to signal changes or power issues, often face unreliable parameter transmission, leading to parameter loss or misdirection (Chen et al. 2018). In distributed algorithms, packet loss due to unreliable networks can divert the algorithm's iterative direction from the theoretical descent path. In severe cases, it can prevent the algorithm from convergence altogether (Peris, Marquina, and Candela 2011; Peters and Wilkinson 1979). Given upon this, it is a natural and crucial question to us: *Can we design a distributed Newton method which is robust and communication*

Copyright © 2024, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

efficient under unrealiable network condition?

We resolve this question by proposing a Packet loss **Re**silient **D**istributed Approximate Newton method **RED-New**. The algorithm transmits inexact approximate directions instead of full Hessian matrices, which effectively reduces the communication scale from $O(d^2)$ to $O(d)$ and enhances the update's resilience to packet loss. This is geared towards fulfilling the demands of practical applications of DML systems. Our main contributions in this paper are as follows:

- We consider the challenge of unreliable network when applying the second-order distributed optimization, which is a common concern in the real-world distributed machine learning systems.
- We propose **RED-New**, which is the first second-order method that can achieve both communication efficiency and resilience to packet loss. We prove a linear quadratic rate which is faster than first-order methods in the local region.
- Our experiments demonstrates the advantage of **RED-New** over second-order and first-order baselines on various packet loss and datasets, validating the theoretical results.

The remainder of this paper is structured as follows: Section 2 delves into existing works aimed at solving the two aforementioned challenges. Our **RED-New** algorithm is detailed in Section 3, with its theoretical convergence bound clarified in Section 4. Section 5 showcases experimental results to demonstrate the efficacy and efficiency of our method. Finally, Section 6 concludes the paper.

Related Work

In this section, we will conduct an in-depth review of related works to identify our unique contributions within this context.

Distributed Second-order Methods

In classic distributed Newton method, each round of updating begins with clients computing local gradients and second-order partial derivatives of the objective function, also known as Hessian matrices, which are then transmitted to a central node. This node aggregates these parameters to calculate the Newton direction, which is the product of the inverse of global Hessian matrix and the global gradient. Following this, it updates the global model and dispatches this updated model back down to the individual clients. The update rule for the classic Newton method in DML is outlined in (2), where $\mathbf{w}_t$ signifies the model at the t-th iteration (Beck 2014). While this method lessen the total communication rounds, it also introduces a substantial number of transmitted bits during each iteration, as it involves sending local gradients, local Hessian matrices, and the globally updated model.

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \left(\sum_{i=1}^{m} \nabla^2 f_i(\mathbf{w}_t)\right)^{-1} \sum_{i=1}^{m} \nabla f_i(\mathbf{w}_t). \quad (2)$$

Research such as **FedNL** (Safaryan et al. 2022; Islamov, Qian, and Richtarik 2021) has proposed solutions to the issue of large data volume transmission in distributed second-order methods. They have extended compressors used in first-order methods to the second-order context, discovering that using contractive compressors like Top-k and Rank-r can be more communication efficient than unbiased compressors like random sparsification and quantization. However, these methods still transmit data at an $O(d^2)$ scale. Other works such as **FedNew** and **Cocoa** (Elgabli et al. 2022; Smith et al. 2018) has developed a two-layer method that cuts the transmitted volume to $O(d)$. The outer layer transforms the global update direction of the second-order Newton method into a distributed constrained quadratic problem. The inner layer then uses distributed first-order methods to solve the problem on clients, thus obtaining an approximate update direction. While this brings the updated direction closer to the standard Newton direction, it also introduces an extra iteration calculation process. Alternatively, methods like **GIANT** (Wang et al. 2018), **DINGO** (Crane and Roosta 2019), and **MNM** (Cao and Lai 2020) limit the computation of the inverse Hessian matrix to local clients. These methods utilize local Hessian matrices to calculate the update direction, and the parameter server employs the local approximate Newton update direction for model updates. While these algorithms are straightforward to compute and memory-friendly, it's crucial to ensure the similarity between the approximate direction and the global Newton direction.

DML Over Unreliable Network

In this subsection, we explore Distributed Machine Learning (DML) in the context of unreliable networks. A common issue in such environments is packet loss, as evidenced by previous studies (Sheth et al. 2007; Lv, Chen, and Wang 2021; Tang, Zhou, and Kato 2020). While the reliable TCP transmission protocol can prevent packet loss, it can also increase network delay significantly with higher packet loss rates (Ye, Liang, and Li 2022). Several research efforts have devised effective solutions by modeling the packet loss network. For instance, the **RSP** approach proposed in (Yu et al. 2019) offers a packet loss resistant first-order algorithm using the AllReduce distributed architecture. It assumes a uniform packet loss probability and substitutes lost parameter blocks with the client's own parameters. However, it overlooks the potential effect of this substitution on the aggregating weight in its theoretical analysis. On the other hand, (Ye, Liang, and Li 2022) accounted for packet loss's impact on the decentralized federal learning architecture, modeling the packet loss probability as a parameter associated with device transmission distance, which aligns more closely with real packet loss scenarios. However, this research's decentralized structure is a fully connected topological structure, and in their theoretical analysis, each bag only contained a single float_32 parameter.

From a systems perspective, additional solutions have been proposed (Chen et al. 2021; Hu et al. 2021). Some works model packet loss as a Gilbert model and employed the network anti-packet loss strategies FEC (Forward Er-

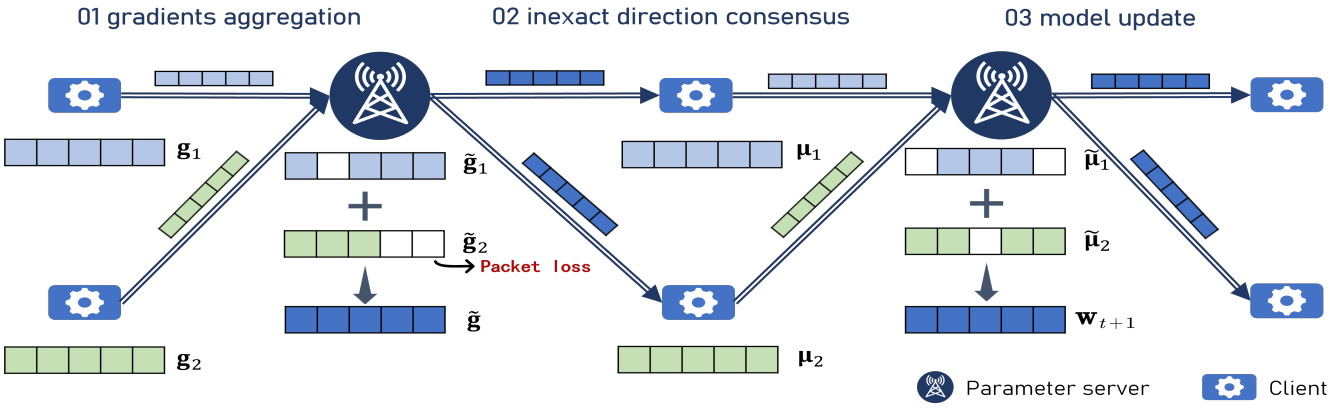


Figure 1: Pipeline operations of RED-New

ror Coding) and ARQ (Automatic Repeat-reQuest). This approach allows the receiver to recover lost data packets when the packet loss rate is not high, thereby lessening the impact of packet loss on the distributed algorithm (Su et al. 2023). Other works such as (Zhang and Tao 2021) involves power control, modeling the unstable network as a signal attenuation transmission model. It optimizes the theoretical difference between the objective function of the inaccurate model and the optimal model through iterative dual algorithms. This technique, despite requiring extra parameter transmission and iterative calculations, is commonly employed in practice.

According to prior discussions, the classic distributed Newton method is impacted more significantly by unreliable networks compared to first-order methods due to its dependence on the inverse of the Hessian matrix. To the best of our knowledge, we have found that research addressing this particular scenario remains sparse, necessitating urgent development of potential solutions. In this study, we specifically design our **RED-New** algorithm to balance both communication efficiency and resilience to packet loss. A solid theoretical analysis supports our approach. Our algorithm shows a superior convergence performance, surpassing traditional gradient descent, even in networks of a high packet loss rate of 40%.

Algorithm Description

In this section, we present the mathematical model of distributed learning over unreliable channels, and our algorithm, namely **RED-New**, catering for such imprecision.

Problem Description

In this study, we address the optimization of problem (1) within the framework of the parameter server (PS) architecture (Li et al. 2014). This paradigm usually consists of one or more servers for synthesizing the global model, and a number of devices as Clients for generating local models.

Model of Packet-loss: This study employs a model that captures the unreliable nature of a network at the packet-level, specifically focusing on packet-loss. For clarity, a diagram

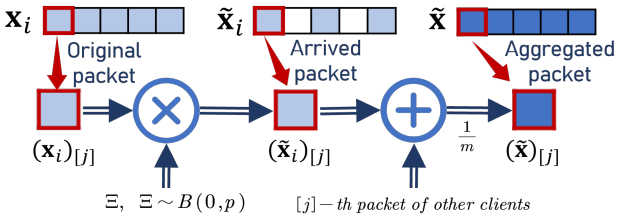


Figure 2: An illustration for the model of packet loss

illustrating our model is provided in Fig 2. Consider a parameter vector, $\mathbf{x}_i \in \mathbb{R}^d$, intended for transmission from the i-th client to the server. Initially, this vector is divided into s blocks. Each block is then contained within a packet, with a maximum size limit of 1500 bytes, as shown in the equation:

$$\mathbf{x}_i = [(\mathbf{x}_i)_{[0]}, (\mathbf{x}_i)_{[1]}, \cdots, (\mathbf{x}_i)_{[s]}]. \tag{3}$$

The j-th block of $\mathbf{x}_i$, denoted as $(\mathbf{x}_i)_{[j]}$, will then be transmitted from the client to the server under an unreliable network.

Noted that the server is usually collocated with a base station or an access point that has large transmission power, and thus the received signal quality at the downlink clients is satisfactory. On the other hand, the clients are power-limited mobiles or Internet of Thing (IoT) devices so that the received signal quality at the server is relatively weak. Therefore, we can reasonably assume that the downlink transmission is reliable and the uplink transmission is prone to packet losses. Our model suggests that packet loss occurs with a uniform probability p. Based on this, if we send $(\mathbf{x}_i)_{[j]}$ over the unreliable network, the arrived packet at the server side, $(\tilde{\mathbf{x}}_i)_{[j]}$, is expressed as:

$$(\tilde{\mathbf{x}}_i)_{[j]} = \Xi((\mathbf{x}_i)_{[j]}) := \begin{cases} (\mathbf{x}_i)_{[j]} & \text{with probability } p \\ \mathbf{0} & \text{with probability } 1-p \end{cases}. \tag{4}$$

Reflecting real-world conditions, we have designated the packet loss rate between 0% and 40%, in line with models from (Chen et al. 2021; Vargaftik et al. 2022).

Upon receiving these packets, the server typically aggregates the parameters to derive a global parameter, denoted as $\tilde{\mathbf{x}}$. From the packet-level perspective, the j-th block of aggregated parameter, $\tilde{\mathbf{x}}_{[j]}$ can be represented as $\tilde{\mathbf{x}}_{[j]} = \frac{1}{m}\sum_{i=1}^{m}(\tilde{\mathbf{x}}_i)_{[j]}$. Unlike this, our **RED-New** proposed a scaling strategy with the use of statistical loss rate to aggregate the parameters, introduced in the following subsection.

RED-New Algorithm

To address the issue of packet loss, **RED-New** employs a packet-level empirical loss rate, denoted as $\hat{p}_{[j]}$. This rate is used to scale the received parameters by a factor of $\frac{1}{1-\hat{p}_{[j]}}$. Consider that the i-th client transmits the vector $\mathbf{x}_i$ to the server through a lossy network, and the arrived packets is $\tilde{\mathbf{x}}_i$, with the j-th packet represented as $(\tilde{\mathbf{x}}_i)_{[j]}$, defined in (4). The empirical loss rate for the subsequent aggregated block, $(\hat{p}_{\mathbf{x}})_{[j]}$, can be expressed as:

$$(\hat{p}_{\mathbf{x}})_{[j]} = \frac{\sum_{i=1}^{m}\mathbb{I}\{(\tilde{\mathbf{x}}_i)_{[j]} = 0\}}{m}, \tag{5}$$

where $\mathbb{I}\{(\tilde{\mathbf{x}}_i)_{[j]} = 0\}$ indicates the count of lost packets and the received j-th block of $\mathbf{x}_i$ is zero.

Using this statistical loss rate, local blocks can be aggregated as:

$$\tilde{x}_{[j]} = \frac{\sum_{i=1}^{m}(\tilde{\mathbf{x}}_i)_{[j]}}{m(1-(\hat{p}_{\mathbf{x}})_{[j]})}, \tag{6}$$

In this context, $\tilde{x}_{[j]}$ is the j-th block of the global parameter aggregated by the server. The well-designed scaling factor, $\frac{1}{1-(\hat{p}_{\mathbf{x}})_{[j]}}$, amplifies the aggregated weight of successfully received data packets when packet loss occurs. By leveraging this approach, the gap between the original $\frac{1}{m}\sum_{i=1}^{m}\mathbf{x}_i$ and $\tilde{x}_{[j]}$ can be effectively constrained by the norm of the exact one.

We formally present the the proposed **RED-New** method in Algorithm 1 and illustrate each steps of **RED-New** as follows. **RED-New** operates in three stages: 1) gradient transmission, 2) inexact update direction transmission, and 3) model update.

Gradient Transmission. At this stage, each client calculates its local gradient and divides the parameters into s packets for delivery denoted as $\mathbf{g}_i = [(\mathbf{g}_i^t)_{[0]}, (\mathbf{g}_i^t)_{[1]}, \cdots, (\mathbf{g}_i^t)_{[s]}]$. Each packet might get lost with probability p. For the j-th packet of client i, its reception on the server follows the rule

$$(\tilde{\mathbf{g}}_i^t)_{[j]} = \Xi\left((\mathbf{g}_i^t)_{[j]}\right). \tag{7}$$

These inexact local gradients are then generated to the global gradient as $\tilde{\mathbf{g}}^t = [(\tilde{\mathbf{g}}^t)_{[0]}, (\tilde{\mathbf{g}}^t)_{[1]}, \cdots, (\tilde{\mathbf{g}}^t)_{[s]}]^T$. In paricular, the j-th block can be computed by

$$(\tilde{\mathbf{g}}^t)_{[j]} = \frac{\sum_{i=1}^{m}(\tilde{\mathbf{g}}_i^t)_{[j]}}{m(1-(\hat{p}_{\mathbf{g}}^t)_{[j]})}, \tag{8}$$

where $(\hat{p}_{\mathbf{g}}^t)_{[j]} = \frac{\sum_{i=1}^{m}\mathbb{I}\{(\tilde{\mathbf{g}}_i^t)_{[j]}=0\}}{m}$.

Algorithm 1: RED-New

input : initial local models $\mathbf{w}_i^0$.

for $t = 0, 1, \ldots$ *until terminate* **do**
- **for** $i = 1$ *to* m *in parallel* **do**
  - The i-th client computes local gradient $\mathbf{g}_i^t = \nabla f_i(\mathbf{w}^t)$.
- The server receives $(\tilde{\mathbf{g}}_i^t)$ according to (7) and compute the global gradient:
- $(\hat{p}_{\mathbf{g}}^t)_{[j]} = \frac{\sum_{i=1}^{m}\mathbb{I}\{(\tilde{\mathbf{g}}_i^t)_{[j]}=0\}}{m}$
- $(\tilde{\mathbf{g}}^t)_{[j]} = \frac{\sum_{i=1}^{m}(\tilde{\mathbf{g}}_i^t)_{[j]}}{m(1-(\hat{p}_{\mathbf{g}}^t)_{[j]})}$.
- $\tilde{\mathbf{g}}^t = [(\tilde{\mathbf{g}}^t)_{[1]}, \cdots, (\tilde{\mathbf{g}}^t)_{[s]}]$
- The server send $(\tilde{\mathbf{g}}^t)$ to each clients.
- **for** $i = 1$ *to* m *in parallel* **do**
  - The i-th client computes local Hessian matrix $\mathbf{H}_i^t = \nabla^2 f_i(\mathbf{w}^t)$.
  - The i-th client computes the inexact local direction $(\boldsymbol{\mu}_i^t) = (\mathbf{H}_i^t)^{-1}\tilde{\mathbf{g}}^t$.
- The server receives $(\tilde{\boldsymbol{\mu}}_i^t)$ according to (10) and compute the update direction:
- $(\hat{p}_{\boldsymbol{\mu}}^t)_{[j]} = \frac{\sum_{i=1}^{m}\mathbb{I}\{(\tilde{\boldsymbol{\mu}}_i^t)_{[j]}=0\}}{m}$
- $(\tilde{\boldsymbol{\mu}}^t)_{[j]} = \frac{\sum_{i=1}^{m}(\tilde{\boldsymbol{\mu}}_i^t)_{[j]}}{m(1-(\hat{p}_{\boldsymbol{\mu}}^t)_{[j]})}$.
- $\tilde{\boldsymbol{\mu}}^t = [(\tilde{\boldsymbol{\mu}}^t)_{[1]}, \cdots, (\tilde{\boldsymbol{\mu}}^t)_{[s]}]$
- The server updates the global parameter: $\mathbf{w}^{t+1} = \mathbf{w}^t - \tilde{\boldsymbol{\mu}}^t$.
- The server send $\mathbf{w}^{t+1}$ to each clients.

Return the final model $\boldsymbol{w}$.

Inexact Update Direction Transmission: Each client computes its local Hessian matrix $\mathbf{H}_i^t$ with the local data samples. Using the received global gradient $\tilde{\mathbf{g}}^t$, each node computes the local Newton update direction $(\boldsymbol{\mu}_i^t)$ by using the local Hessian information,

$$(\boldsymbol{\mu}_i^t) = (\mathbf{H}_i^t)^{-1}\tilde{\mathbf{g}}^t. \tag{9}$$

$(\boldsymbol{\mu}_i^t)$ is then divided into s packets and sent to the server, also in the face of possible packet loss Ξ. The server receives the local updated direction according to

$$(\tilde{\boldsymbol{\mu}}_i^t)_{[j]} = \Xi\left((\boldsymbol{\mu}_i^t)_{[j]}\right) \tag{10}$$

and aggregate them as $\tilde{\boldsymbol{\mu}}^t = [(\tilde{\boldsymbol{\mu}}^t)_{[0]}, (\tilde{\boldsymbol{\mu}}^t)_{[1]}, \cdots, (\tilde{\boldsymbol{\mu}}^t)_{[s]}]$. We calculate $(\tilde{\boldsymbol{\mu}}^t)_{[j]}$ as

$$(\tilde{\boldsymbol{\mu}}^t)_{[j]} = \frac{\sum_{i=1}^{m}(\tilde{\boldsymbol{\mu}}_i^t)_{[j]}}{m(1-(\hat{p}_{\boldsymbol{\mu}}^t)_{[j]})}, \tag{11}$$

where $(\hat{p}_{\boldsymbol{\mu}}^t)_{[j]} = \frac{\sum_{i=1}^{m}\mathbb{I}\{(\tilde{\boldsymbol{\mu}}_i^t)_{[j]}=0\}}{m}$.

Model Update: Finally, the parameter server updates the model weights $\mathbf{w}$ based on the global approximate inexact update direction $\boldsymbol{\mu}$ and dispatches them to each client,

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \tilde{\boldsymbol{\mu}}^t \tag{12}$$

RED-New excels the classic distributed Newton method in two aspects. First, the data volume transmitted in one iteration is reduced from $O(d^2)$ matrices to $O(d)$ vectors, significantly reducing the communication overhead. Second, unlike the direct inversion of the lossy Hessian which may leads to a huge deviation from the Newton direction, especially when the Hessian matrix is ill-conditioned. Our algorithm avoid such inversion by limiting the inverse operation of matrix to a local level, thus permitting the application of the concentration inequalities to constrain the gap's upper bound norm to ensure algorithmic stability even under unreliable network conditions. Furthermore, a scaling strategy is employed that doesn't require any prior knowledge of the packet loss rate, ensuring the algorithm adapts effectively to real-world scenarios suffers from unreliable network connections.

Convergence Analysis

In this section, we prove that **RED-New** possesses a linear-quadratic convergence rate, implying that the algorithm performs better than first-order methods in terms of the number of training rounds till convergence.

We first introduce the notations for the analysis. We use $\mathbf{H}_t$ and $\mathbf{g}^t$ to denote $\nabla^2 f(\mathbf{w}_t)$ and $\nabla f(\mathbf{w}_t)$. We use $\|\cdot\|_2$ to present spectral norm and Euclidean norm of matrix and vector respectively. Additionally, we introduce the weighted norm $\|\cdot\|_{\mathbf{H}}$ such that $\|\mathbf{x}\|_{\mathbf{H}} := \sqrt{\mathbf{x}^T\mathbf{H}\mathbf{x}}$. We use the following measure to present the convergence

$$\boldsymbol{\Delta}_t := \|\mathbf{w}_t - \mathbf{w}^*\|_{\mathbf{H}_t}.$$

We then introdice the following standard assumptions on the object function (1).

Assumption 1. *The loss $f(\mathbf{w})$ has L_2-Lipschitz Hessian matrices, i.e., $\left\|\nabla^2 f(\mathbf{x}) - \nabla^2 f(\mathbf{y})\right\|_2 \leqslant L\|\mathbf{x}-\mathbf{y}\|_2$ holds for all $\mathbf{x}$ and $\mathbf{y}$.*

Assumption 2. *The Hessian matrix satisfies the form of $\nabla^2 f(\mathbf{w}) = \mathbf{Q}(\mathbf{w})^T\mathbf{Q}(\mathbf{w}) + \mathbf{B}(\mathbf{w})$, where $\mathbf{B}(\mathbf{w})$ is a positive semi-definite matrix.*

Remark 1. A large range of function class like $f(\mathbf{w}) = l(\mathbf{w}^\top\mathbf{x}) + \lambda\|\mathbf{w}\|^2$ satisfy Assumption 2. Such assumption is also commonly adopted in the literature (Wang et al. 2018; Ye, Luo, and Zhang 2021; Derezinski et al. 2021).

Lemma 1. *Under Assumption 2, if a sample size at each worker node satisfies $l \geqslant \left(\frac{3d\log(dm/\delta)}{\epsilon^2}\right)$, then*

$$(1-\epsilon)\mathbf{H} \preccurlyeq \widetilde{\mathbf{H}_i} \preccurlyeq (1+\epsilon)\mathbf{H},$$
$$\left(1-\frac{\epsilon}{\sqrt{m}}\right)\mathbf{H} \preccurlyeq \widetilde{\mathbf{H}_i} \preccurlyeq \left(1+\frac{\epsilon}{\sqrt{m}}\right)\mathbf{H},$$

holds with probability at least $1-\delta$, $\delta, \epsilon \in (0,1)$.

Remark 3. This lemma states that the local Hessian can be constrained by the scale of the global Hessian with a high probability. It leverages the properties of Sketching Matrices (Drineas, Mahoney, and Muthukrishnan 2006; Clarkson and Woodruff 2017), where the local Hessian can be constructed as $\widetilde{\mathbf{H}_i} = [\mathbf{S}_i\mathbf{Q}(\mathbf{x})]^T[\mathbf{S}_i\mathbf{Q}(\mathbf{x})]$. Here, $\mathbf{S}_i \in \mathbb{R}^{l\times n}$ is a leverage score matrix that is well-designed to satisfy $(1-\epsilon)\mathbf{Q}(\mathbf{x})^T\mathbf{Q}(\mathbf{x}) \leqslant [\mathbf{S}_i\mathbf{Q}(\mathbf{x})]^T[\mathbf{S}_i\mathbf{Q}(\mathbf{x})] \leqslant (1+\epsilon)\ \mathbf{Q}(\mathbf{x})^T\ \mathbf{Q}(\mathbf{x})$. This theoretical result was previously proven in (Pilanci and Wainwright 2017).

Lastly, based on the lemma and assumptions, we present our main theoretical results below.

Theorem 1. *Under Assumption 1 and 2, running Algorithm 1 on the quadratic object function, if the sample size on each workers satisfies that $l \geqslant \left(\frac{3d\log(dm/\delta)}{\epsilon^2}\right)$, it holds that*

$$\|\boldsymbol{\Delta}_t\|_{\mathbf{H}_t} \leqslant (\beta)\|\boldsymbol{\Delta}_{t-1}\|_{\mathbf{H}_t},$$

where

$$\beta = \frac{1}{\sqrt{m}}\cdot\frac{\epsilon}{1+\epsilon} + \frac{C_{\gamma,p}}{\delta(1-\epsilon)} + \frac{C_{\gamma,p}}{\delta^2(1-\epsilon)(1-\gamma)},$$
$$C_{\gamma,p} = 2p + \frac{\gamma}{1-\gamma},\ \gamma = \sqrt{\frac{-2log\delta}{m(1-p)}}.$$

with probability $1-\delta$ and $\delta \in (0,1)$.

Theorem 1 reveals the efficacy of the **RED-New** method on a quadratic loss function, achieving a linear convergence rate. Notably, the use of Hessian norms in our analysis, unlike the norm-2 $\|\boldsymbol{\Delta}_t\|_2$ employed in related works (Pilanci and Wainwright 2017; Wang et al. 2018), leads to stronger theoretical results. Specifically, this approach frees the decay factor from the condition number, eliminating the need for a more strict range on β.

Next, we provide the convergence result for **RED-New** for general object function.

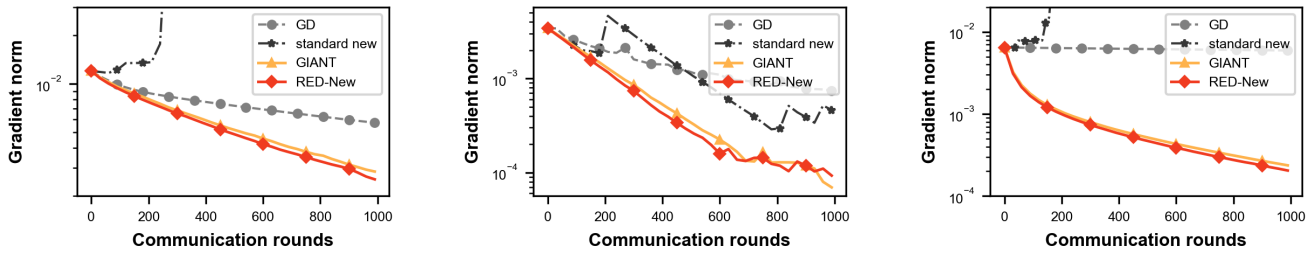
Theorem 2. *Under Assumption 1 and 2, running Algorithm 1 on the general smooth object function, if the sample size on each workers satisfies that $l \geqslant \left(\frac{3d\log(dm/\delta)}{\epsilon^2}\right)$, it holds with at least $1-\delta, \delta \in (0,1)$ that*

$$\|\boldsymbol{\Delta}_{t+1}\|_{\mathbf{H}_t} \leqslant \max\left\{\rho_1\|\boldsymbol{\Delta}_t\|_{\mathbf{H}_t},\ \rho_2\|\boldsymbol{\Delta}_t\|^2_{\mathbf{H}_t}\right\},$$

where,

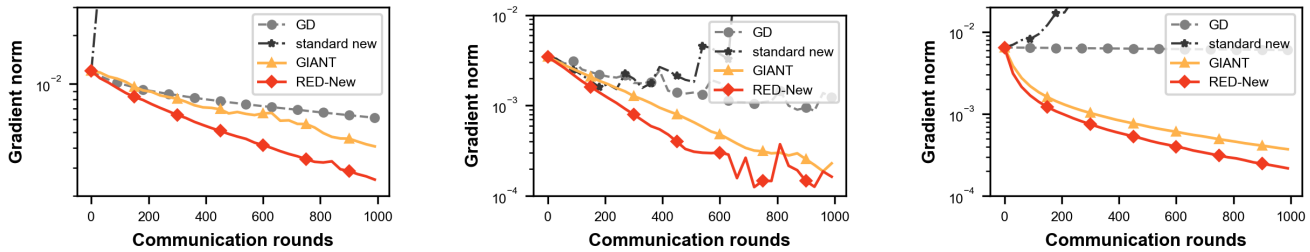
$$\rho_1 = 2\sqrt{\frac{\beta^2}{1-\beta^2}},$$
$$\rho_2 = \frac{2L}{\sigma_{\min}(\mathbf{H}_t)^{\frac{3}{2}}},$$
$$\beta = \frac{1}{\sqrt{m}}\cdot\frac{\epsilon}{1+\epsilon} + \frac{C_{\gamma,p}}{\delta(1-\epsilon)} + \frac{C_{\gamma,p}}{\delta^2(1-\epsilon)(1-\gamma)},$$
$$C_{\gamma,p} = 2p + \frac{\gamma}{1-\gamma},\ \gamma = \sqrt{\frac{-2log\delta}{m(1-p)}}.$$

Theorem 2 provides a local linear-quadratic convergence for **RED-New** on the general smooth loss function. The quadratic term inherits from the classical Newton method that embraces a quadratic convergence rate. The linear term in the inequality stems from the approximation of the Newton direction (Ye, Luo, and Zhang 2021). Theorem 2 shows that the proposed method enjoys a faster rate than the first-order methods especially When ϵ, p, and δ are small.



(a) w8a (b) a9a (c) gisette

Figure 3: Training loss comparison of GD, Standard Newton, GIANT, and RED-New over the unreliable network with a packet loss rate $p = 5\%$ on three datasets, a9a, w8a, and gisette



(a) w8a (b) a9a (c) gisette

Figure 4: Training loss comparison of GD, Standard Newton, GIANT, and RED-New over the unreliable network with a packet loss rate $p = 20\%$ on three datasets, a9a, w8a, and gisette

Experiments

Our experiments are conducted on the benchmark LIBSVM Libraray datasets (Chang and Lin 2011) for the convex problem of logistic regression with l_2-regularization:

$$\min_{\mathbf{w}} \frac{1}{n} \sum_{j=1}^{n} \log\left(1 + \exp\left(-y_i {x_i}^T \mathbf{w}\right)\right) + \frac{\lambda}{2} \|\mathbf{w}\|_2^2, \quad (13)$$

where $x_i \in \mathbb{R}^d$ is the feature vector and $y_i \in \{-1, 1\}$ is the corresponding label. The regularization coefficient λ controls the generalization ability, and is set to 10^{-6}. We conduct our experiments using Python with the Mpy4pi 3.1.4 function for distributed optimization. The experiments are operated on the Intel(R) Xeon(R) Platinum 8369B 2.90GHz, equipped with 32 CPU cores and 2.0TB memory.

Experimental Setting

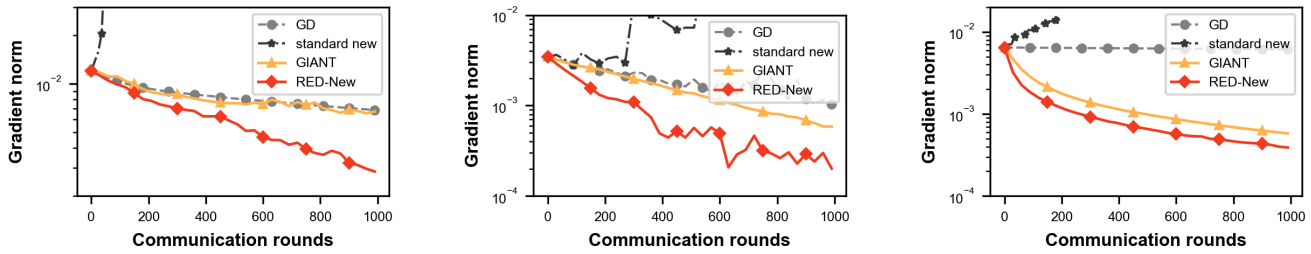
Parameter configuration: We employ three commonly used datasets from the LIBSVM Library: a9a ($N = 32561, d = 123$), w8a ($N = 49749, d = 300$), and gisette ($N = 6000, d = 5000$). Here, N denotes the number of samples and d is the number of features of each sample. We set 8 workers for the distributed training and a warm-up strategy is used. To make our experiments more general, we introduce random features based on the RBF kernel, expanding the number of features to 800 for a9a, and 1000 for w8a.

During transmission, each client encapsulates 375 float_32 data into a packet of 1500 Bytes. Equivalently, at each iteration of two communication rounds, the number of packet transmitted by a client will be 6 for a9a and w8a, 32 for gisette.

Our baseline methods include the distributed gradient descent, the standard distributed Newton method, and the approximate distributed Newton method GIANT (Wang et al. 2018). The learning rate of the proposed methods and the baselines are tuned from $\{1, 0.1, 0.01, 0.001\}$.

Experimental Results

Comparison of Training loss: To evaluate the performance of **RED-New**, we conduct experiments with different packet loss rate p that ranges from 0 to 40%, and compare the results with other baselines. Fig.3$\sim$ 5 illustrate the training losses of different algorithms against the number of training rounds with p set to 5%, 20%, and 40% respectively. One can observe that both **RED-New** and GIANT converge much faster than the first-order gradient descent method, and the standard Newton method does not converge even the packet loss rate is merely 5%. This throws light on the limitations of certain algorithms in challenging network en-



(a) w8a (b) a9a (c) gisette

Figure 5: Training loss comparison of GD, Standard Newton, GIANT, and RED-New over the unreliable network with a packet loss rate $p = 40\%$ on three datasets, a9a, w8a, and gisette

Traffic(KB)	p**=5%**				p**=20%**				p**=40%**			
	a9a	**w8a**	**gisette**	**Redu.**	**a9a**	**w8a**	**gisette**	**Redu.**	**a9a**	**w8a**	**gisette**	**Redu.**
GD	3931	2449	12343	*89.07%*	4668	2875	14726	*90.77%*	6246	3902	19512	*92.33%*
STNEW	958640	∞		99.97%	∞			–	∞			–
GIANT	356	664	39	*6.47%*	500	945	78	*40.63%*	887	1555	117.19	*61.69%*
RED-New	325	593	39	–	331	585	39	–	337	679	39	–

Table 1: Comparison of Traffic Volume (KB) for Various Algorithms reaching the predefined objective value $\hat{f}$. Results are shown for three datasets: a9a, w8a, and gisette. The Redu. item represents average reduction ratio contributed by RED-New in Traffic volume compared to other benchmark algorithms.

vironments, as the inversion of Hessian is quite susceptible to the impact of inaccurate transmission. When the packet loss rate becomes large, **RED-New** considerably outperforms GIANT in terms of the number of training rounds till convergence. This is especially evident when compared to the robust performance of our **RED-New** algorithm. The experiment outcomes align with the foundational intuition behind **RED-New**. While it adopts an approximation direction similar to GIANT, it extends further to a higher robustness with an scaling strategy adaptive to the real-world packet loss scenarios. This aligns with our goal of developing a second-order algorithm that is both communication-efficient and robust, demonstrating the resilience of our method in the face of the unreliable network with packet loss.

Comparison of Traffic volume and Training time:

Table 1 measures the uplink traffic volume until the objective function reaches the predefined tolerance $\hat{f}$. For fair evaluation, $\hat{f}$ of a given dataset is selected as the largest value of a series of minimum objectives attained by different algorithms after a fixed number of training rounds.

The traffic volume for each iteration is calculated based on the number of parameters transmitted uplink to the server, interpreting a float_32 datum as equivalent to 4 bytes. The empirical results reveal the notable efficiency of the **RED-New** algorithm in curtailing traffic volume. Specifically, when faced with a lossy environment characterized by a 40% packet loss rate, **RED-New** reduces traffic by an average of 92.33% compared to the first-order method, and 61.69% in comparison to GIANT. Comparing across varying packet loss rate, such reduction in communication volume becomes more remarkable as the packet loss rate increasing. An illustrative example of this is the comparative reduction ratio against GIANT. In scenarios with a 5% lossy network, **RED-New** achieves a traffic volume decrement of 6.47%. This ratio soars to a substantial 61.69% when faced with a packet loss rate of 40%. These numerical results highlight the advantage of **RED-New** in terms of reduced traffic volume, especially as packet loss rates escalate.

Conclusions

In this paper, we propose **RED-New**, a second-order Newton method that is both robust to the packet loss and communication efficient. We prove that over unreliable network, our algorithm can reach an improved linear-quadratic convergence rate. Numerical experiments are conducted to evaluate the convergence rate of **RED-New**, significantly surpassing the first-order and second-order baselines over same packet loss rate, ranging from 5% to 40%. In the future, it will be interesting to investigate the trade-off between per-iteration communication efficiency and the total number of training rounds, utilizing Forward Error Correction (FEC) to achieve an optimal balance.

Acknowledgments

This work was supported in part by the Natural Science Foundation of China under Grant 62072117, the Shanghai Natural Science Foundation under Grant 22ZR1407000, and the Huawei project on New Network Transport Layer. Dr. Yuedong Xu is the corresponding author for this research.

References

Beck, A. 2014. *Introduction to Nonlinear Optimization*. Philadelphia, PA: Society for Industrial and Applied Mathematics.

Boyd, S. 2011. Convex Optimization: From Embedded Real-Time to Large-Scale Distributed. In *KDD*, KDD '11, 1. New York, NY, USA: Association for Computing Machinery. ISBN 9781450308137.

Cao, X.; and Lai, L. 2020. Distributed Approximate Newton's Method Robust to Byzantine Attackers. *IEEE Transactions on Signal Processing*, 68: 6011–6025.

Chang, C.-C.; and Lin, C.-J. 2011. LIBSVM: A Library for Support Vector Machines. *ACM Trans. Intell. Syst. Technol.*, 2(3).

Chen, L.; Zhao, N.; Chen, Y.; Yu, F. R.; and Wei, G. 2018. Over-the-Air Computation for IoT Networks: Computing Multiple Functions With Antenna Arrays. *IEEE Internet of Things Journal*, 5(6): 5296–5306.

Chen, M.; Gündüz, D.; Huang, K.; Saad, W.; Bennis, M.; Feljan, A. V.; and Poor, H. V. 2021. Distributed Learning in Wireless Networks: Recent Progress and Future Challenges. *IEEE Journal on Selected Areas in Communications*, 39(12): 3579–3605.

Clarkson, K. L.; and Woodruff, D. P. 2017. Low-Rank Approximation and Regression in Input Sparsity Time. *J. ACM*, 63(6).

Crane, R.; and Roosta, F. 2019. DINGO: Distributed Newton-Type Method for Gradient-Norm Optimization. *Advances in Neural Information Processing Systems*, 32.

Derezinski, M.; Lacotte, J.; Pilanci, M.; and Mahoney, M. W. 2021. Newton-LESS: Sparsification without Trade-offs for the Sketched Newton Update. *Advances in Neural Information Processing Systems*, 34: 2835–2847.

Ding, Y.; Niu, C.; Wu, F.; Tang, S.; Lyu, C.; feng, y.; and Chen, G. 2022. Federated Submodel Optimization for Hot and Cold Data Features. *Advances in Neural Information Processing Systems*, 35: 1–13.

Drineas, P.; Mahoney, M. W.; and Muthukrishnan, S. 2006. Sampling Algorithms for L2 Regression and Applications. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm*, SODA '06, 1127–1136. USA: Society for Industrial and Applied Mathematics. ISBN 0898716055.

Elgabli, A.; Issaid, C. B.; Bedi, A. S.; Rajawat, K.; Bennis, M.; and Aggarwal, V. 2022. FedNew: A Communication-Efficient and Privacy-Preserving Newton-Type Method for Federated Learning. In Chaudhuri, K.; Jegelka, S.; Song, L.; Szepesvari, C.; Niu, G.; and Sabato, S., eds., *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, 5861–5877. PMLR.

Hu, S.; Chen, X.; Ni, W.; Hossain, E.; and Wang, X. 2021. Distributed Machine Learning for Wireless Communication Networks: Techniques, Architectures, and Applications. *IEEE Communications Surveys & Tutorials*, 23(3): 1458–1493.

Islamov, R.; Qian, X.; and Richtarik, P. 2021. Distributed Second Order Methods with Fast Rates and Compressed Communication. In Meila, M.; and Zhang, T., eds., *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, 4617–4628. PMLR.

Konečný, J.; McMahan, H. B.; Yu, F. X.; Richtárik, P.; Suresh, A. T.; and Bacon, D. 2017. Federated Learning: Strategies for Improving Communication Efficiency. arXiv:1610.05492.

Lee, J. D.; Lin, Q.; Ma, T.; and Yang, T. 2017. Distributed Stochastic Variance Reduced Gradient Methods by Sampling Extra Data with Replacement. *Journal of Machine Learning Research*, 18(122): 1–43.

Li, M.; Andersen, D. G.; Smola, A. J.; and Yu, K. 2014. Communication Efficient Distributed Machine Learning with the Parameter Server. In Ghahramani, Z.; Welling, M.; Cortes, C.; Lawrence, N.; and Weinberger, K., eds., *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Li, X.; Huang, K.; Yang, W.; Wang, S.; and Zhang, Z. 2020. On the Convergence of FedAvg on Non-IID Data. arXiv:1907.02189.

Liu, C.; Chen, L.; Luo, L.; and Lui, J. C. S. 2023. Communication Efficient Distributed Newton Method with Fast Convergence Rates. arXiv:2305.17945.

Lv, Z.; Chen, D.; and Wang, Q. 2021. Diversified Technologies in Internet of Vehicles Under Intelligent Edge Computing. *IEEE Transactions on Intelligent Transportation Systems*, 22(4): 2048–2059.

McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and Arcas, B. A. y. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Singh, A.; and Zhu, J., eds., *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, 1273–1282. PMLR.

Mishchenko, K.; Malinovsky, G.; Stich, S.; and Richtarik, P. 2022. ProxSkip: Yes! Local Gradient Steps Provably Lead to Communication Acceleration! Finally! In Chaudhuri, K.; Jegelka, S.; Song, L.; Szepesvari, C.; Niu, G.; and Sabato, S., eds., *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, 15750–15769. PMLR.

Mitra, A.; Jaafar, R.; Pappas, G. J.; and Hassani, H. 2021. Linear Convergence in Federated Learning: Tackling Client Heterogeneity and Sparse Gradients. *Advances in Neural Information Processing Systems*, 34: 14606–14619.

Nesterov, Y. 2007. Accelerating the Cubic Regularization of Newton's Method on Convex Problems. *Math. Program.*, 112(1): 159–181.

Nesterov, Y.; and Polyak, B. T. 2006. Cubic Regularization of Newton Method and Its Global Performance. *Math. Program.*, 108(1): 177–205.

Peris, R.; Marquina, A.; and Candela, V. 2011. The convergence of the perturbed Newton method and its applica-

tion for ill-conditioned problems. *Applied Mathematics and Computation*, 218(7): 2988–3001.

Peters, G.; and Wilkinson, J. H. 1979. Inverse Iteration, Ill-Conditioned Equations and Newton's Method. *SIAM Review*, 21(3): 339–360.

Pilanci, M.; and Wainwright, M. J. 2017. Newton Sketch: A Near Linear-Time Optimization Algorithm with Linear-Quadratic Convergence. *SIAM Journal on Optimization*, 27(1): 205–245.

Safaryan, M.; Islamov, R.; Qian, X.; and Richtarik, P. 2022. FedNL: Making Newton-Type Methods Applicable to Federated Learning. *Proc. of the 39th International Conference on Machine Learning*, 162: 18959–19010.

Shamir, O.; Srebro, N.; and Zhang, T. 2014. Communication-Efficient Distributed Optimization using an Approximate Newton-type Method. In Xing, E. P.; and Jebara, T., eds., *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, 1000–1008. Bejing, China: PMLR.

Sheth, A.; Nedevschi, S.; Patra, R.; Surana, S.; Brewer, E.; and Subramanian, L. 2007. Packet Loss Characterization in WiFi-Based Long Distance Networks. In *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, 312–320.

Smith, V.; Forte, S.; Ma, C.; Takáč, M.; Jordan, M. I.; and Jaggi, M. 2018. CoCoA: A General Framework for Communication-Efficient Distributed Optimization. *Journal of Machine Learning Research*, 18(230): 1–49.

Su, X.; Zhou, Y.; Cui, L.; and Liu, J. 2023. On Model Transmission Strategies in Federated Learning With Lossy Communications. *IEEE Transactions on Parallel and Distributed Systems*, 34(4): 1173–1185.

Tang, F.; Zhou, Y.; and Kato, N. 2020. Deep Reinforcement Learning for Dynamic Uplink/Downlink Resource Allocation in High Mobility 5G HetNet. *IEEE Journal on Selected Areas in Communications*, 38(12): 2773–2782.

Vargaftik, S.; Basat, R. B.; Portnoy, A.; Mendelson, G.; Itzhak, Y. B.; and Mitzenmacher, M. 2022. EDEN: Communication-Efficient and Robust Distributed Mean Estimation for Federated Learning. In Chaudhuri, K.; Jegelka, S.; Song, L.; Szepesvari, C.; Niu, G.; and Sabato, S., eds., *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, 21984–22014. PMLR.

Wang, S.; Roosta, F.; Xu, P.; and Mahoney, M. W. 2018. GIANT: Globally Improved Approximate Newton Method for Distributed Optimization. *Advances in Neural Information Processing Systems*, 31.

Ye, H.; Liang, L.; and Li, G. Y. 2022. Decentralized Federated Learning With Unreliable Communications. *IEEE Journal of Selected Topics in Signal Processing*, 16(3): 487–500.

Ye, H.; Luo, L.; and Zhang, Z. 2021. Approximate Newton Methods. *The Journal of Machine Learning Research*, 22.

Yu, C.; Tang, H.; Renggli, C.; Kassing, S.; Singla, A.; Alistarh, D.; Zhang, C.; and Liu, J. 2019. Distributed Learning over Unreliable Networks. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, 7202–7212. PMLR.

Zhang, N.; and Tao, M. 2021. Gradient Statistics Aware Power Control for Over-the-Air Federated Learning. *IEEE Transactions on Wireless Communications*, 20(8): 5115–5128.