DISTRIBUTED LEARNING AND INFERENCE IN DEEP MODELS

A Dissertation
Presented to
The Academic Faculty

By

Afshin Abdi

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering
College of Engineering

Georgia Institute of Technology

August 2020

DISTRIBUTED LEARNING AND INFERENCE IN DEEP MODELS

Thesis committee:

Professor Faramarz Fekri, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Matthieu Bloch
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Ghassan AlRegib
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Siva Theja Maguluri
School of Industrial and Systems Engineering
Georgia Institute of Technology

Professor Justin Romberg
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date approved: July 20, 2020

To my family

for their encouragement and supports.

ACKNOWLEDGMENTS

I would like to express my gratitude to my research advisor prof. Faramarz Fekri for his kindness, continuous support, and advice. His encouragement and mentorship have always helped me a lot during the course of my Ph.D. research. I would also like to especially thank my thesis committee members: prof. Justin Romberg, prof. Ghassan AlRegib, prof. Siva Theja Maguluri, and prof. Matthieu Bloch, who were more than generous with their expertise and precious time.

Many thanks to my close friend and collaborator Alireza Aghasi whom I have learned a lot from. The discussions we had and his suggestions and advice have greatly helped me. I would also like to thank my close friend, Ali Payani, for his priceless supports and guidance since the first day of my Ph.D. at Georgia Tech. My sincere thanks go to my colleagues Yashas Malur Saidutta and Hang Zhang for the invaluable discussions and collaborations we had on various projects. Last but not least, I would also like to thank my other friends and collaborators at Georgia Tech: Dr. Entao Liu, Dr. Xin Tian, Dr. Sohail Bahmani, and Dr. Ahmad Beirami.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Recent breakthroughs in artificial intelligence and machine learning, as well as the availability of large datasets, are transforming many aspects of daily life, businesses, and industries. On the other hand, the complexity of deep learning problems has increased significantly in recent years. Hence, training or real-time inference of modern deep models on end-user devices or a single processing node is unappealing or nearly impossible due to the required storage, memory or computational power. The overarching theme of my Ph.D. thesis is addressing the challenges raised in deep learning due to the complexity of models. Especially, we consider communication bottleneck in distributed training and inference of deep models.

Distributed Deep Learning

One of the main challenges in distributed training is the communication cost due to the transmission of the parameters or stochastic gradients (SGs) of the deep model for synchronization across processing nodes (a.k.a. workers). Compression is a viable tool to mitigate the communication bottleneck. However, the existing methods suffer from a few drawbacks, such as increased variance of SG, slower convergence rate, or added bias to SG. In my Ph.D. research, we have addressed these challenges from three different perspectives:

1. *Information Theory and the CEO Problem*: we argued that the computations at each worker can be considered as noisy observations of the true update (or gradient), $\boldsymbol{\theta}^*$, and the objective of distributed learning would be reliable estimation of $\boldsymbol{\theta}^*$ with minimum communication from workers. We use this principle to develop a framework for efficient data communication in distributed learning.
2. *Matrix Factorization*: One way of compressing the SGs is low-rank matrix factorization and quantization. However, naively pursuing such an approach is costly in distributed machine learning, in terms of the computations and the training error. By exploiting the factorization inherent in the backpropagation algorithm, quantizer optimization, and

controlled dithering, we develop two novel Indirect SG Quantization (ISGQ) methods. ISGQ is unbiased, and with the same number of quantization levels, it has lower MSE and computational complexity than most SG compression methods.

3. *Compressive Sampling*: The performance and compression gain of ISGQ are limited by the structure of neural networks. To achieve arbitrarily large unbiased compression of SG, we considered projecting SG into a small random subspace, and then compressing it. Inspired by the structured random mixing matrices and utilizing controlled dithering and quantization, we developed Quantized Compressive Sampling (QCS). We showed that QCS is unbiased and can achieve orders of magnitude smaller MSE than other unbiased compression methods, resulting in superior convergence rate.

Next, we consider federated learning over wireless multiple access channels (MAC). Efficient communication requires the compression algorithm to satisfy the constraints imposed by the nodes in the network, communication channel, and data privacy. To satisfy these constraints and take advantage of the over-the-air computation inherent in MAC, we propose a framework based on random linear coding and develop efficient power management and channel usage techniques to manage the trade-offs between power consumption, communication bit-rate, and convergence rate of federated learning.

Model Restructuring and Adjustment for Distributed Inference

While the complexity of modern deep neural networks allows them to learn complicated tasks, the computational complexity and memory footprint limit their usage in many real-time applications as well as deployment on many end-user devices with limited resources. Hence, model reduction and adjustment is a highly desirable process for deep neural networks. In the second part of my thesis, we consider the distributed parallel implementation of an already-trained deep model on multiple workers. As such, the deep model is divided into several parallel sub-models, each of which is executed by a worker. Since latency due to synchronization and data transfer among workers negatively impacts the performance of the parallel implementation, it is desirable to have minimum interdependency among parallel

sub-models. To achieve this goal, we develop and analyze RePurpose, an efficient algorithm to rearrange the neurons in the neural network and partition them (without changing the general topology of the neural network) such that the interdependency among sub-models is minimized under the computations and communications constraints of the workers.

CHAPTER 1
INTRODUCTION AND LITERATURE SURVEY

In recent years, the size of deep learning problems has been increased significantly, both in terms of the number of available training samples as well as the number of parameters and complexity of the model. On the other hand, the limited RAM memory capacity and computational power of a single processing unit make training of modern deep models challenging. Moreover, the increased inference time of large models makes them unfavorable for real-time applications such as virtual assistants and autonomous vehicles. In this thesis, we consider the challenges encountered in training and inference of large deep models, especially on nodes with limited computational power and capacity. We consider two classes of related problems; 1) distributed training of deep models, and 2) compression and restructuring of deep models for efficient distributed and parallel inference to reduce execution times on devices and networks with limited resources.

1.1 Distributed Data Training

Availability of large datasets is one of the main deriving forces for the recent surge in the applications of deep models. However, in practice, transferring all data to a central powerful node may be infeasible due to (i) the dataset is too large to be stored in a single node, (ii) the data is inherently distributed, or (iii) moving data is expensive or prohibited due to the privacy concerns. Hence, training deep models on a single processing node can be unappealing or nearly impossible. On the other hand, as the complexity of deep models increases (i.e., number of parameters or layers), massive amounts of storage, memory, and computational power are required for training deep models in a reasonable amount of time. As such, large-scale distributed machine learning in which the training samples are distributed among different repository or processing units (referred to as workers) has

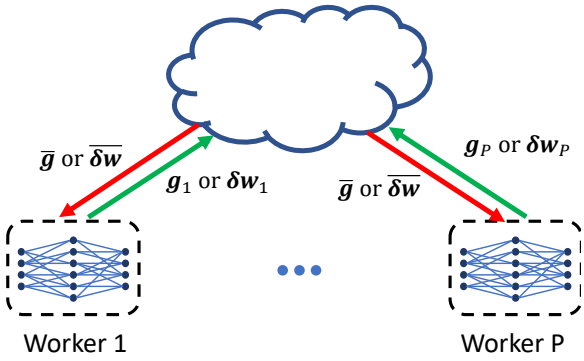


Figure 1.1: Overview of distributed data training

started to become a viable approach for tackling the challenges in complex deep learning problems [1, 2, 3, 4, 5, 6, 7, 8].

In the first part of the thesis, we consider the problem of deep learning when training data is distributed. We assume that each worker has access to its own training data (a subset of the entire training samples) and uses its training samples to locally update the model. The models at the workers are then synchronized by exchanging information (such as stochastic gradients or updates of model's parameters) between workers and a parameter server in the *centralized* distributed training, or among workers in the *decentralized* setting. Ideally, it is expected that by increasing the number of workers and hence the total computational power, the training time would be decreased proportionately. However, as the scale of distributed systems grows large, the extensive information exchanges for model synchronization across workers incur significant communication overhead. The resulting communication over the limited channel bandwidth increases in the total training time in practice.

In recent years, there has been a great amount of effort on mitigating the communication bottleneck in distributed training. Most of these methods can be applied to both centralized and decentralizes settings although we will present them in the context of centralized training for simplicity. The existing methods can be summarized as follows:

1. **Quantization**: Reducing the number of bits in representing SG (or parameter updates) is a well-known technique to decrease the communication bit-rate. For example, [9]

suggested quantizing the gradients to 1-bit by mapping positive values to τ_+ and negative ones to τ_-. The reconstruction points τ_+ and τ_- are found via minimizing the mean squared error of the quantizer. They showed that the 1-bit quantization scheme can significantly reduce the communication overhead without any major loss in the accuracy of the final trained model, as long as the quantization error is carried forward to the next mini-batch. However, the reduced accuracy of gradients and quantization bias may impair the convergence rate. Using different quantization levels and/or adaptive quantizers, one can alleviate such issues [10, 11, 12]. Other techniques such as SignSGD [13, 14] are also proposed to directly use only the sign of the gradients, $\{-1, +1\}$, for the optimization. One major drawback of using ordinary (deterministic) quantization methods is the added bias to the stochastic gradients. To ensure the convergence of the training algorithm with biased SG, it is crucial to incorporate the error feedback during quantization. An alternative approach to avoid the quantization bias is using random (stochastic) quantization, i.e., $\widehat{\boldsymbol{g}}$ is a random variable such that $\mathbb{E}[\widehat{\boldsymbol{g}}] = \boldsymbol{g}$. QSGD [15] and TernGrad [16] are examples of such approaches which guarantee the convergence of the training algorithm without using error feedback and provide a trade-off between the gradient precision and the model accuracy. However, the reduced precision of SG due to the quantization error can potentially increase the training time.

2. **Sparsification**: Another approach to reduce the communication overhead is transmitting only the important or a small subset of the gradients. [17] was among the early works to use sparsification in conjunction with thresholded quantization to further compress the gradients; it compacts and threshold gradients whose magnitude exceed a certain value. As choosing the right threshold for gradient sparsification is difficult in practice, other approaches have been proposed such as transmitting only a fixed portion of the gradients [10, 18, 19, 20], TopK SGD [21, 22], deep gradient compression [23], and sparse communication [24]. Since generally the sparsification

results in biased stochastic gradients, it is crucial to aggregate the residuals to ensure the convergence of the learning algorithm. In parallel, [25] has proposed random (stochastic) sparsification and scaling of the gradients to achieve both sparsity and unbiasedness. A similar approach, Atomo [26], considers the sparsification of the gradients in the transform domain such that the variance of error is minimized subject to a given average sparsity budget.

3. **Using Error Feedback**: Application of quantization or sparsification techniques in deep learning may introduce two major issues: (i) increase in the variance of the aggregated gradients, and (ii) insertion of a bias to the stochastic gradient. These may degrade the convergence speed or even cause the learning algorithm fail to converge. A key component in tackling both of these issues is aggregating the compression residuals (i.e., quantization or sparsification errors) and carrying forward to the next mini-batch. This ensures that the true values of SG are eventually applied to the parameters of the deep model, although it may take several transmissions. Exploiting such a feedback can speed up the convergence rate or ensure the convergence of the learning algorithms such as stochastic gradient descents even in the presence of (biased) gradient compression [19, 27, 28]. However, it is worth noting that storing the residuals increases the memory footprint of the algorithm proportional to the size of the deep model which might be undesirable in some applications, especially for large models. Moreover, since adding the residual to the gradients can potentially increase the variance of the values, it is important to adjust the learning parameters accordingly to avoid divergence of the algorithm.

4. **Entropy Coding**: The outcome of quantization/sparsification steps may be followed by simple entropy coding algorithms such as adaptive arithmetic coding or ad-hoc compression techniques to further reduce the communication overhead [29, 15].

5. **Stale Synchronous and Asynchronous Training**: Finally, another group of works

attempts to reduce the communication bottleneck by relaxing the synchronization between workers [30, 31, 32, 33]. Each worker may continue its own computations while some others are still communicating and exchanging parameters. Carefully scheduling and managing the asynchronous parameter exchange can lead to a better utilization of both the communication bandwidth and the computational power of the distributed system [34, 35, 36, 37]. Examples of such approaches include DownpourSGD [38] and Stale Synchronous Parallel model of computation [39, 40, 41]. Hogwild! [31] and Hogwild++ [42] allow the workers to access a shared memory with possibility of overwriting each other's work. It is shown that when most gradient updates are sparse, it achieves a nearly optimal rate of convergence.

Another technique to minimize the communication among workers is synchronizing the models among workers only occasionally. Local SGD is based on running SGD independently in parallel on different workers and averaging the parameters of the model only once in a while. [43, 44, 45] have shown that this scheme can converge at the same rate as mini-batch SGD.

Throughout chapters 2, 3, and 4 of the thesis, we assume the synchronous distributed training scheme, and focus on improving the training speed via reducing the communication bit-rate or improving the variance of the compression algorithm for faster convergence rates.

In chapter 2, we argue that the deep model's parameters or stochastic gradients can be highly correlated among different workers; providing opportunity for distributed compression [46]. Moreover, the main objective in distributed learning is a good estimate of the model's parameters at the server by using the information received from workers, rather than the exact recovery of model's updates (or gradients) from each individual worker. Hence, we frame the distributed training as the Central Estimation Officer (CEO) problem [47, 48, 49]. To reduce the communication overhead, we model the dependency among parameters computed by the workers and propose different distributed compression algorithms to take advantage of that correlation [50, 51].

In chapter 3, we consider factorization of SG into smaller matrices and then compressing the factorized terms for SG compression. However, naively pursuing such an approach is costly in distributed machine learning, in terms of both computational complexity and the training error. By exploiting the factorization inherent in the backpropagation algorithm, quantizer optimization, and controlled dithering, we develop Indirect SG Quantization (ISGQ) method. ISGQ is unbiased, and with the same number of quantization levels, it has lower MSE and computational complexity than most SG compression methods, which translates into faster convergence rates [52].

In chapter 4, we investigate the problem of obtaining arbitrarily large unbiased compression gains while ensuring that the mean squared error (MSE) of the compressed SG is low. To achieve this goal, we consider projecting SG into a small random subspace, and then compressing it. Inspired by the structured random mixing matrices and utilizing controlled dithering and quantization, we develop Quantized Compressive Sampling (QCS). We show that QCS is unbiased and can achieve orders of magnitude smaller MSE than other unbiased compression methods, resulting in superior convergence rate. Moreover, we develop weighted error feedback and analyze how it can reduce the gap in the convergence rate compared to the baseline [53].

Finally, we consider federated learning over wireless multiple-access-channels in chapter 5. Federated learning differs from traditional distributed machine learning as 1) the data observed by the nodes are usually unbalanced and non-iid, and 2) all nodes may not transmit at every round of communications. Hence, distributed optimization algorithms which are often developed for high performance computing clusters are not readily applicable to federated learning [54, 55, 56, 57, 58, 59]. We develop an efficient communication algorithm that satisfies the constraints imposed by the communication medium and take advantage of its characteristics, such as over-the-air computations inherent in wireless multiple-access channels (MAC) [60, 61], unreliable transmission and idle nodes in the the network, limited transmission power, and preserving the privacy of data [62].

1.2 Parallel and Distributed Inference

In recent years, the size and complexity of deep neural networks has been increased significantly in terms of model's structure and number of parameters. Consequently, real-time implementation and inference in many machine learning (ML) problems has become a challenging task. Although the execution time of deep neural networks can be improved significantly by the application of parallel computing algorithms and using multiple processing units (such as GPU's or clusters of computing nodes), it generally requires synchronization and data exchange among processing units to some extent. This is mainly due to the fact that in parallel computations, each processing unit performs a portion of the computations, its inputs generally depend on the outputs from other units, and the results of computations should be aggregated to yield the desired output. These co-dependencies can lead to significant delays in computations. For example, in a GPU, accessing the shared memory within a block of threads has lower latency compared to accessing the global shared memory. Moreover, synchronization among separate blocks of threads can lead to idle processing times and lower computing efficiency. Hence, it is more desirable to run blocks of threads independently. Moreover, in some real-world scenarios, such as sensor networks, the inference is done on the data observed by the entire network, i.e., each node in the network only observes a portion of the input data. However, transferring all data to a central powerful node to aggregate and perform the ML task is undesirable due to the sheer amount of data to be collected, limited computational power, privacy concerns, or even availability of such a node. Hence, it is more favorable to develop a distributed equivalence of a deep model for deploying over the processors/sensor network. As such, the network, as a whole, would become a computing engine of the original deep model for inference from data observed by all nodes.

In the aforementioned applications, straightforward parallel computing algorithms cannot be arbitrarily scaled up for deep models with complex connectivity structures. As such,

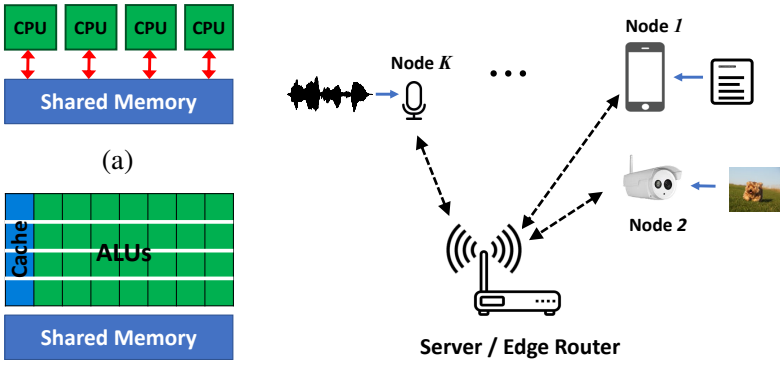


Figure 1.2: Examples of parallel distributed model, (a) a multicore system, (b) GPUs, (c) distributed model over a sensor network

there is an increasing interest in efficient parallel execution, reducing complexity of deep models, or modifying their structure for nearly optimal deployment with only subtle changes in their performance.

The majority of past works on distributed/parallel execution of deep neural networks are concerned with algorithmic aspects of the parallel implementation of the neural network (e.g., [1, 63, 64]). However, in chapter 6, we focus on the structure of deep models and how we can modify it for efficient parallel distributed implementation. The majority of these approaches can be classified into three categories:

1. **Knowledge Distillation**: In knowledge distillation, the goal is to train a shallow or small model (referred to as student network) that mimics the behavior of an already trained complex model (referred to as teacher network) or an ensemble of teacher networks [65]. Using soft targets instead of the hard labels has the benefits of preventing the student model from being too sure during training, and allowing each training data to impose more constraints on the weights. Extensions of such approaches include Fitnets [66] and Attention transfer [67]. However, one major drawback of knowledge distillation is that it can only be applied to classification tasks with softmax output.

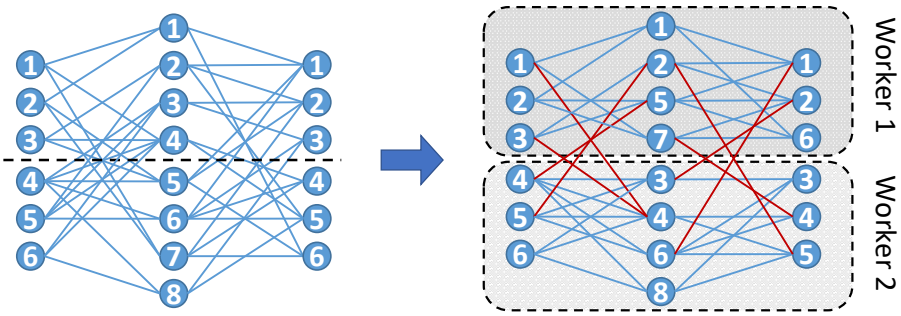
2. **Using Structured Parameters** to reduce the size of deep model or its processing

time. Examples include using circulant matrices [68] or Adaptive Fastfood transform [69] for fully connected layers, and separable filters [70] or low-rank tensor decomposition [71] for convolutional layers.

3. **Reducing Parameters via Quantization, Pruning and Clustering**: Network pruning has been used to reduce the complexity of the model as well as to address the over-fitting. ℓ_1 regularization, group-sparsity [72, 73] or ℓ_0 [74] can promote sparsity of the parameters during training. Network pruning algorithms such as the Optimal Brain Damage [75], the Optimal Brain Surgeon [76], hard-thresholding the parameters [77], and similar works [78, 79], mainly focus on removing the insignificant edges or nodes to reduce the size of the model. They generally consider the magnitude of the weight or an approximation to the Hessian matrix as a measure of the importance. Alternatively, Net-Trim, [80, 81], uses a convex optimization technique to prune the parameters of the deep model by analyzing the signals in the neural network.

   Another approach to reduce the memory footprint of deep models is via quantizing the parameters and using fewer bits. [82] used vector quantization to compress the parameters of CNN. [83] quantizes the parameters of a pre-trained model and proposes an optimization technique for training fixed point deep CNNs. Higher compression gains can be achieved via layer-specific quantization levels. On the extreme case, some works, such as Binaryconnect [84], Binarized neural networks [85], TBN [86], Trained Ternary quantization [87], and XNOR-Net [88], try to directly train a 1-bit or Ternary deep neural network.

Although it is possible to design deep models according to the capability and constraints of the processing system, following such an approach requires training a new deep model for every target hardware which is infeasible or demanding in many ML problems. Further, imposing a possibly unnecessary structure in advance during training a deep model would likely be limiting in terms of model performance and accuracy. It will be also an undesirable



(a) Original model (b) Restructured model

Figure 1.3: Restructuring a neural network to reduce communication between processing units

approach for parallel implementation since a model specifically designed for optimum implementation on a target platform or architecture may be far from optimum on other platforms (e.g., GPUs with different compute capabilities, or CPU vs GPU vs sensor network). Hence optimizing and fixing the structure for one particular parallel distributed setting in advance would limit the optimal deployment on other platforms. As a result, in [89], we assume that a complex deep model has already been trained with minimum or no hardware-specific constraints on its parameters or structure. Our goal would be readjusting the model via restructuring the layers and manipulating the parameters of the neural network without changing its general topology for more efficient parallel implementation. Without changing the general topology of the neural network, we propose to rearrange the neurons and partition the deep model into sub-models with minimum co-dependency subject to the computation and communication constraints on the workers.

As an example, consider the simple neural network in Fig. 1.3(a). Simply partitioning the model into two sub-models (as depicted by a dashed line in the Fig. 1.3(a)) imposes lots of communication between the two partitions. However, by rearranging the neurons properly, the co-dependency (and hence required communications) between the two sub-models (the red edges in Fig. 1.3(b)) is reduced substantially. It is worth mentioning that there are approximately $\mathcal{O}(P^N)$ different partitioning to distribute computations of a neural network's layer with N neurons over P workers. Hence, enumerating all such possibilities

and choosing a good one is infeasible specially for large networks. In chapter 6, we propose a systematic approach to perform such partitioning and parameter adjustment to ensure efficient implementation of the modified model while keeping its accuracy close to the original model.

1.3 Notations

Bold lowercase letters represent vectors and the i-th element of the vector $\boldsymbol{x}$ is denoted as x_i. Matrices are denoted by bold capital letters such as $\boldsymbol{X}$, with the (i, j)-th element represented by $X_{i,j}$ or $[\boldsymbol{X}]_{i,j}$, the i-th row by $\boldsymbol{X}_{i,\cdot}$ and the j-th column by $\boldsymbol{X}_{\cdot,j}$ or $\boldsymbol{x}_j$. $\boldsymbol{A} \odot \boldsymbol{B}$ is the Hadamard product of $\boldsymbol{A}$ and $\boldsymbol{B}$. $\boldsymbol{A} \odot \boldsymbol{v}$ for vector $\boldsymbol{v}$ is computed by expanding the dimension of $\boldsymbol{v}$ appropriately to make it the same size as $\boldsymbol{A}$.

Given a real number $x \in \mathbb{R}$, $\lfloor x \rfloor$ is the largest integer smaller than or equal to x, $\lceil x \rceil$ the smallest integer greater than or equal to x and $\lfloor x \rceil$ represents the nearest integer to x. $\mathrm{sign}(x)$ is the sign of x defined as $+1$ for $x > 0$ and -1 for $x \leq 0$. $\log$ and $\log_2$ denote the natural and base 2 logarithms, respectively.

A Normal distribution with mean μ and variance σ^2 is denoted by $\mathcal{N}(\mu, \sigma^2)$. Likewise, $\mathcal{U}(a, b)$ is uniform distribution over interval (a, b).

CHAPTER 2
DISTRIBUTED TRAINING AND THE CEO PROBLEM

2.1 Introduction

One of the main challenges in distributed training is the communication cost due to the transmission of the parameters or stochastic gradients of the model and synchronization across processing nodes (a.k.a. workers). To mitigate the communication bottleneck, many works have considered reducing the transmission bit-rate by compressing the stochastic gradients (or parameter updates) via techniques such as quantization, sparsification, and entropy coding. However, these works have some drawbacks: 1) the existing methods do not leverage the redundancy in the information transmitted by different workers. Model's parameters at different workers are highly correlated, which provides an opportunity for distributed compression to reduce the communication bit-rate, and 2) the main objective in distributed learning is a good estimate of the model's parameters at the server by using the information received from workers, rather than the exact recovery of model's updates (or gradients) from each worker. For example, in distributed stochastic gradient computation, the objective is computing the average of the stochastic gradients computed by all workers, not recovering the exact value of the SG of each individual worker.

In this chapter, we address the above shortcomings of existing methods by leveraging information theoretic tools, especially compression of correlated sources and coding for function computations.

2.2 Problem Statement

Consider the problem of distributed optimization of a cost function $\mathcal{J}(\boldsymbol{w}) = \mathbb{E}_{\boldsymbol{x}\in\mathcal{X}}[f(\boldsymbol{x};\boldsymbol{w})]$, where $\mathcal{X}$ is the whole training dataset and $f(\boldsymbol{x};\boldsymbol{w})$ measures the error in fitting the model

determined by parameters $\boldsymbol{w}$ to the input data point $\boldsymbol{x}$. In this chapter, we focus on the synchronous distributed training where a central node, a.k.a. the server, is responsible for aggregating the parameters/information from the workers, computing a shared model update and broadcasting it back to the workers.

We consider the following general framework at each iteration of distributed training:

1. The server broadcasts its initial guess of the model's parameters to all workers.

2. Each worker *refines* its local copy of the parameters by training over its available dataset and then sends back the new parameters (or the updates) to the server.

3. The server merges the received information and estimates a better parameter for the deep model.

At an arbitrary iteration t of the distributed training algorithm with P workers, let $\boldsymbol{w}_0 \in \mathbb{R}^N$ be the model's parameter, shared by all workers and the server.[1] The p-th worker uses its available local data and updates the parameters to $\boldsymbol{w}_p$ which it believes is a better solution to minimize the objective function $\mathcal{J}(\boldsymbol{w})$. Ideally, if all data were available at a single node, we would expect to arrive at the updated parameter $\boldsymbol{w}^*$ for the next iteration of the learning algorithm. However due to access to only a subset of the training dataset, the estimate of the optimum parameters by the p-th worker would be *noisy*. Hence, we model the updated parameters at the p-th worker by

$$\boldsymbol{w}_p = \boldsymbol{w}^* + \boldsymbol{n}_p, \tag{2.1}$$

or alternatively,

$$\boldsymbol{\theta}_p = \boldsymbol{\theta}^* + \boldsymbol{n}_p, \tag{2.2}$$

where $\boldsymbol{\theta}^* = \boldsymbol{w}^* - \boldsymbol{w}_0$, $\boldsymbol{\theta}_p = \boldsymbol{w}_p - \boldsymbol{w}_0$ is the amount of update in the parameters by the p-th worker, and $\boldsymbol{n}_p$ is the parameter estimation noise at the p-th worker, assumed to have mean

[1] Note that $\boldsymbol{w}_0$ and the parameters computed by the workers and the server depend on the iteration t. However, for the sake of simplicity, we omitted such dependencies in our the notations.

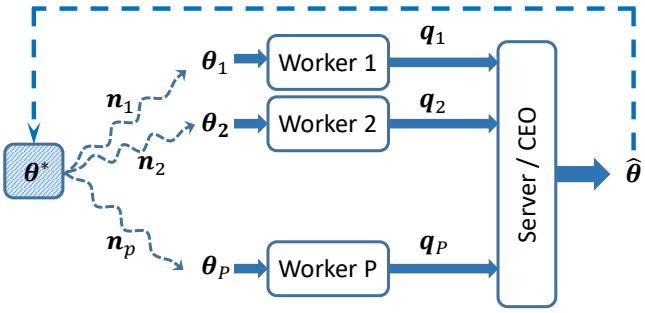


Figure 2.1: Distributed training as the CEO problem

zero and variance $\sigma_p^2 := \mathbb{E}[\|\boldsymbol{n}_p\|^2]$. The objective is estimating $\boldsymbol{\theta}^*$ from the $\boldsymbol{\theta}_p$'s computed by the workers such that the estimation error of $\boldsymbol{\theta}^*$ is minimized subject to a given constraint on the average communication bits. Hence, the distributed learning can be viewed as a Central Estimating Officer (CEO) problem, well-known in information theory [47, 48, 90]; each worker has a *noisy* observation of an *unknown* variable and wants to compress and transmit it to the server, from which the server can estimate the optimum parameter reliably and broadcasts back to the workers for the next iteration of training (Fig. 2.1).

Remark 1. For distributed computation of the stochastic gradients, we can simply assume that $\boldsymbol{\theta}_p$ is the stochastic gradient (SG) computed by the p-th worker over its mini-batch and $\boldsymbol{\theta}^*$ is the *true* gradient of the cost function. Assuming that the noises in (2.2) are i.i.d. Gaussian, the best estimate of the true gradient would be the average of all workers' SGs. Hence, the objective of the CEO in the distributed training would be minimizing the communication rate for computing the average of SGs from the workers.

2.3 Distributed Learning as the CEO Problem

Recall that the distributed training can be modeled as the CEO problem where each worker observes a noisy replica of the desired variable $\boldsymbol{\theta}^*$ as

$$\boldsymbol{\theta}_p = \boldsymbol{\theta}^* + \boldsymbol{n}_p, \tag{2.3}$$

where $\boldsymbol{n}_p$ is a zero-mean noise with $\sigma_p^2 := \mathbb{E}[\|\boldsymbol{n}_p\|^2]$. Inspired by coding for the CEO problem [48], the proposed communication scheme for the distributed training consists of the following building blocks:

1. Quantization: Each worker p quantizes $\boldsymbol{\theta}_p$ to a sequence $\boldsymbol{q}_p$ such that the distortion is within a given bound.

2. Compression: The workers use appropriate (distributed or entropy source) coding to compress the quantized sequences and transmit them to the server.

3. Decoding and Estimation: The server decodes the received sequences and estimates $\boldsymbol{\theta}^*$ by $\hat{\boldsymbol{\theta}}$ such that $\mathbb{E}\left[\|\boldsymbol{\theta}^* - \hat{\boldsymbol{\theta}}\|_2^2\right]$ is minimized.

In the following, we elaborate more on each of the aforementioned steps.

2.3.1 Quantization at the Workers

It is well-known that the error in ordinary (deterministic) quantization depends on the input signal, especially when the number of quantization levels is low. This can adversely affect the convergence of distributed training algorithms. As such, we propose using *dithered quantization* or *nested quantization* of the parameters at the workers.

Dithered Quantization

In dithered quantization, an independent (pseudo-)random *dither* signal is added to the input signal prior to the quantization and is subtracted after dequantization. By carefully controlling the properties of the dither signal, one can achieve the desired statistical behavior of the quantized values.

Definition (Dithered Quantization)**.** Let ϱ be the quantization step size. For an input signal x, assume that u is a random dither signal, generated independently of x. The dithered quantization of x is defined as $\tilde{x} = \varrho(\lfloor x/\varrho + u \rceil - u)$, where $\lfloor \alpha \rceil$ is the nearest integer to α.

Remark 2. To transmit the dithered quantization of x, it is sufficient to send the index of the quantization bin that $x/\varrho + u$ resides in, i.e., $q = \lfloor x/\varrho + u \rceil$. The receiver can reproduce the (pseudo-)random sequence u using the same random number generator algorithm and seed number and then compute the quantized value as $\tilde{x} = \varrho(q - u)$.

Characteristics of the dither signal has a major impact on the properties of the quantization noise. It is known that if the dither signal is generated uniformly over $(-1/2, 1/2)$, i.e. $u \sim \mathcal{U}(-1/2, 1/2)$, then the quantization noise $e = x - \tilde{x}$ is independent of the signal x and $e \sim \mathcal{U}(-\varrho/2, \varrho/2)$.

We consider the following dithered quantization at the p-th worker; Let M be the number of quantization levels, $\kappa_p = \|\boldsymbol{\theta}_p\|_\infty / M$ be the scale factor, and $\boldsymbol{u}_p \sim \mathcal{U}(-1/2, 1/2)$ be a random dither signal generated independently by the p-th worker. The dithered quantization at worker p is given by

$$\boldsymbol{q}_p = \left\lfloor \boldsymbol{u}_p + \frac{\boldsymbol{\theta}_p}{\kappa_p} \right\rceil, \tag{2.4}$$

where $\lfloor x \rceil$ is the closest integer to x.

To reconstruct $\boldsymbol{\theta}_p$, the server has to generate the same random sequence $\boldsymbol{u}_p$ and receive the scale factor κ_p in addition to the quantized values $\boldsymbol{q}_p$.[2] The dequantized value at the server is then computed as

$$\widetilde{\boldsymbol{\theta}}_p = \kappa_p(\boldsymbol{q}_p - \boldsymbol{u}_p). \tag{2.5}$$

Note that the range of the quantized values would be $\{-M, \ldots, 0, \ldots, +M\}$. Further, as a result of [91], the scaled quantization error $(\boldsymbol{\theta}_p - \tilde{\boldsymbol{\theta}}_p)/\kappa_p$ would be independent of $\boldsymbol{\theta}_p$ and uniformly distributed over $(-1/2, 1/2)$.

[2]To generate the same dither signal, the server and worker are both initialized to the same seed value for random number generation. By using the same random number generation algorithm and updating the seed number in the same manner, both can generate the same dither $\boldsymbol{u}_p$. Here, we do not consider any quantization or compression for the transmission of the scale factors as their communication overhead is negligible.

Nested Quantization

It is well-known that correlated signals can be communicated more efficiently via distributed compression than the traditional entropy coding algorithms [46]. Nested Quantization has been proven to be a viable tool in distributed compression [92] when a correlated side information is available at the receiver. Here, we briefly overview a variant of the nested quantization in one-dimension which our proposed distributed training is based on it.

Consider the problem of transmitting x where a side information y is available at the receiver. Let (Q_1, Q_2) be a pair of nested quantizers with quantization step sizes Δ_1 and Δ_2, respectively, i.e., $Q_i(v) = \Delta_i \lfloor v/\Delta_i \rceil$ for $i = 1, 2$. [3] To quantize and transmit x, the transmitter first generates a random dither $u \sim \mathcal{U}(-\Delta_1/2, \Delta_1/2)$ and computes $t = x + u$. Then t is quantized and encoded as

$$s = Q_1(t) - Q_2(t), \tag{2.6}$$

i.e., it transmits the position of the fine quantization bin relative to the coarse one (shown by indexes $-1, 0, 1$ in Fig. 2.2). At the receiver, by knowing s alone, x cannot be estimated reliably as multiple values can produce the same s. To resolve that ambiguity, it is required to know which coarse quantization bin x belongs to. This is achieved by the help of the side information y, available at the receiver. In this case, the estimated x is computed as follows:

$$r = s - u - y, \quad \hat{x} = y + r - Q_2(r). \tag{2.7}$$

Note that quantizing x does not require y, however estimating x at the server depends on the information provided by y. Figure 2.2 shows an example of using nested quantization, where $\Delta_1 = 1$ and $\Delta_2 = 3$, $x = -4.2$, $y = -3.4$ and $u = 0.3$. Therefore, the nested quantized value would be $s = -1$. Note that many points can produce the same s, some are shown

[3] For our purposes, to have a pair of nested quantizers, it suffices to have $\Delta_2 = k\,\Delta_1$ where $k > 1$ is an integer number.

by ♦. However, having access to y at the receiver can resolve that ambiguity, resulting in $\hat{x} = -4.3$.

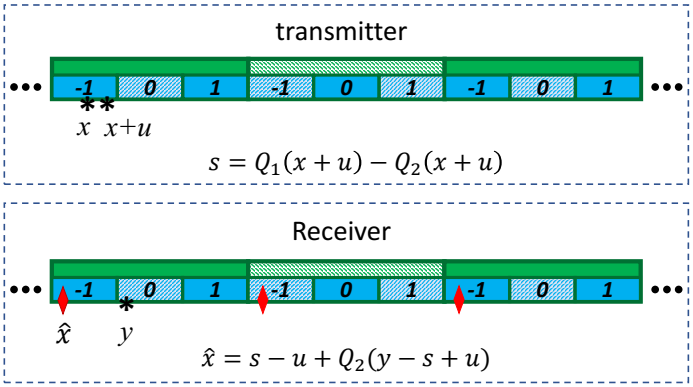


Figure 2.2: Example of nested quantization, $\Delta_1 = 1$ and $\Delta_2 = 3$. For $x = -4.2$ and $u = 0.3$, the nested quantized value would be $s = -1$. Note that many points can produce the same s, some are shown by ♦. However, having access to $y = -3.4$ at the receiver can resolve that ambiguity, resulting in $\hat{x} = -4.3$.

Applying the above quantization scheme for distributed training requires having access to a correlated side-information. This is achieved via dividing the workers into two groups: the first group, $\mathcal{P}_1$, uses ordinary dithered quantization whose transmitted data is used to generate the required side information for nested quantization, and the second group, $\mathcal{P}_2$, uses the above nested quantization scheme for the quantization and compression of their parameters. Let $\bar{\boldsymbol{\theta}}$ be the side information computed using data received from workers in $\mathcal{P}_1$, e.g., by dequantizing and averaging the received values. The nested quantization uses $\bar{\boldsymbol{\theta}}$ at the server as the side information to compute $\widetilde{\boldsymbol{\theta}}_p$ for worker p in the second group. We assume that the parameters of the p-th worker can be modeled as $\boldsymbol{\theta}_p = \bar{\boldsymbol{\theta}} + \boldsymbol{z}_p$, where $\boldsymbol{z}_p$ is an independent random noise. The following lemma bounds the probability of error and variance of the quantization error;

Lemma 1. *If the parameters at a worker is modeled by $\boldsymbol{\theta} = \bar{\boldsymbol{\theta}} + \boldsymbol{z}$, $\mathbb{E}[\|\boldsymbol{z}\|^2] = \sigma_z^2$, and the worker uses nested quantizer with step-sizes Δ_1, Δ_2, then with probability at least $1 - p$, $\tilde{\theta}$, the nested quantization of θ, will be estimated correctly (i.e., the distance between θ and $\tilde{\theta}$*

would be less than Δ_2), where

$$p = \Pr\left(|z+u| > \frac{\Delta_2}{2}\right) \leq \frac{\Delta_1^2}{3\Delta_2^2} + 4\frac{\sigma_z^2}{\Delta_2^2}, \tag{2.8}$$

for $u \sim \mathcal{U}[-\Delta_1/2, \Delta_1/2)$. Specially if $|\boldsymbol{z}| < \frac{\Delta_2 - \Delta_1}{2}$, then $p = 0$. In this case,

$$\mathbb{E}\left[\|\widetilde{\boldsymbol{\theta}} - \boldsymbol{\theta}\|_2^2\right] = \frac{\Delta_1^2}{12}. \tag{2.9}$$

Note that nested quantization results in the same quantization variance as dithered quantization with step-size Δ_1. However, nested quantization requires $\log_2(\Delta_2/\Delta_1)$ bits to transmit each value, less than the ordinary quantization methods which require almost $\log_2(2/\Delta_1)$ bits.

2.3.2 Compression

To further reduce the communication bit-rate, each worker p applies source coding to the quantized sequence $\boldsymbol{q}_p$. There are two possible approaches: 1) Simple entropy coding algorithms such as adaptive arithmetic coding, and 2) more complex distributed source coding methods. Since $\boldsymbol{\theta}_p$'s are correlated, there is a dependency among the quantized sequences $\boldsymbol{q}_p$'s and distributed source coding (DSC) algorithms [46] can be used to further reduce the communication bit rate. We propose using DISCUS [93] based on LDPC codes because of its simple 'encoding' algorithm at the workers and reliable decoding using message passing at the server. For the simplicity, we use asymmetric DSC; the workers are divided into two groups, $\mathcal{C}_1$ ad $\mathcal{C}_2$. Workers in $\mathcal{C}_1$ use entropy coding to compress their data. The decoded sequences from these workers serve as side-information at the server for decoding the data from the workers in $\mathcal{C}_2$ which use DISCUS. Moreover, the joint probability distribution The decoded sequence from these workers serve as side information for decoding the data of the next worker. Let $\hat{\boldsymbol{\theta}}_1$ be the estimated $\boldsymbol{\theta}^*$ using received data from

workers in $\mathcal{C}_1$. The conditional distribution of the sequences from $\mathcal{C}_1$ w.r.t. $\hat{\boldsymbol{\theta}}_1$ is considered as an estimate of the conditional pdf of the sequences from $\mathcal{C}_2$ and used at the server for decoding.

Knowing the compression rates at the workers is enough to use the appropriate DISCUS encoder. Therefore, few encoders of various rates are pre-designed and deployed at the workers. The appropriate encoder is determined by the server via statistical analysis of the quantized values received from the workers in $\mathcal{C}_1$.

2.3.3 Estimating $\boldsymbol{\theta}^*$

Recall that the quantization noise, $\boldsymbol{e}_p = \widetilde{\boldsymbol{\theta}}_p - \boldsymbol{\theta}_p$, in the dithered quantization is independent of the signal and uniformly distributed[4] with variance $\eta_p^2 := \mathbb{E}[\|\boldsymbol{e}_p\|^2] = N\kappa_p^2\Delta_1^2/12$, where N is the number of the parameters ($\boldsymbol{\theta}_p \in \mathbb{R}^N$), $\kappa_p = \|\boldsymbol{\theta}_p\|_\infty$ and Δ_1 is the quantization step-size. Hence,

$$\widetilde{\boldsymbol{\theta}}_p = \boldsymbol{\theta}^* + \boldsymbol{\nu}_p, \tag{2.10}$$

where $\boldsymbol{\nu}_p = \boldsymbol{e}_p + \boldsymbol{n}_p$, $\mathbb{E}[\|\boldsymbol{\nu}_p\|_2^2] = \eta_p^2 + \sigma_p^2$.

As the prior distribution on $\boldsymbol{\theta}^*$ is not known, we consider minimax linear MMSE estimator.

Lemma 2. *The weights of the minimax MMSE linear estimator $\hat{\boldsymbol{\theta}}_P = \sum_{i=1}^P \alpha_i \widetilde{\boldsymbol{\theta}}_i$ are given by*

$$\alpha_i = \frac{1}{\gamma_p \ \mathbb{E}[\|\boldsymbol{\nu}_i\|_2^2]}, \tag{2.11}$$

where γ_p is chosen such that $\sum_{i=1}^p \alpha_i = 1$.

The following recursive equations computes $\hat{\boldsymbol{\theta}}_p$'s for $p = 1, \ldots, P$:

[4]Here, we ignored the dependency on the scaling factor κ_p as it is treated separately in our distributed system.

$$\gamma_1 = 1/\,\mathbb{E}\big[\|\boldsymbol{\nu}_1\|_2^2\big], \quad \hat{\boldsymbol{\theta}}_1 = \widetilde{\boldsymbol{\theta}}_1 \tag{2.12a}$$

$$\gamma_{p+1} = \frac{1}{\mathbb{E}[\|\boldsymbol{\nu}_{p+1}\|_2^2]} + \gamma_p \tag{2.12b}$$

$$\hat{\boldsymbol{\theta}}_{p+1} = \left(\gamma_p\, \hat{\boldsymbol{\theta}}_p + \frac{\widetilde{\boldsymbol{\theta}}_{p+1}}{\mathbb{E}[\|\boldsymbol{\nu}_{p+1}\|_2^2]}\right)/\gamma_{p+1} \tag{2.12c}$$

Note that the above linear estimator is also Minimax estimator if the total noise in (2.10) follows Gaussian distribution.

2.4 Experiments

For our experiments, we have considered different models, a fully connected neural network with two hidden layers of sizes 300 and 100 (herein, referred to as FC-300-100) and a Lenet-5 like convolutional network [94] over MNIST, as well as a convolutional network on Cifar10 [95] (referred to as CifarNet). We have used stochastic gradient descent and Adam training algorithms. The initial learning rates are 0.01 with decay rate 0.9 per training epoch. The batch size is fixed at 200 and divided evenly among the workers. We compare our proposed communication methods against the baseline (no quantization of gradients), QSG [15] with the same quantization accuracy, and one-bit quantization [9] for different number of workers. For fair comparison, we apply entropy coding to the quantized sequences of these methods as well.

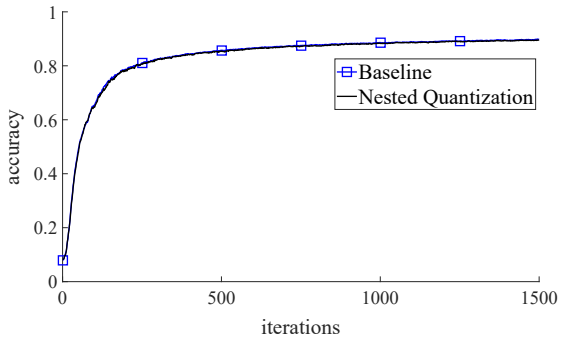
First, we compare the performance of nested quantization followed by adaptive arithmetic coding to the other existing methods. For nested quantization, we divided the workers in half, the first group uses ordinary dithered quantization and the second group uses nested quantization with $(\Delta_1, \Delta_2) = (1/3, 1)$. All workers use adaptive arithmetic coding (AAC) for further compression of the quantized values. Table 2.1 shows the raw (un-compressed) and compressed communication bits per worker at each iteration of training for different neural networks and 16 workers. It is worth mentioning that almost the same communication bit-rate was observed for distributed training using $2, 4, 8, \ldots$ workers. Note that although

the raw communication bit-rate of one-bit quantization [9] is less than the others, it is less compressible which results in almost the same compressed communication bits per worker. On the other hand, as observed by others as well, the low accuracy of one-bit quantization adversely affects the convergence speed and requires more iterations of the distributed training algorithm. Hence, generally one-bit quantization results in more overall communication for the convergence of the distributed learning algorithm. Moreover, the requirement to store the quantization error and carry it forward to the next mini-batch imposes additional memory requirement which is unappealing esp. for sensor networks. Figure 2.3 shows the convergence rate of the proposed nested quantization scheme compared with the baseline (no quantization of the parameters) using SGD training algorithm. As seen from the plots, the proposed method perfroms closely to the raw (unquantized) transmission of parameters.

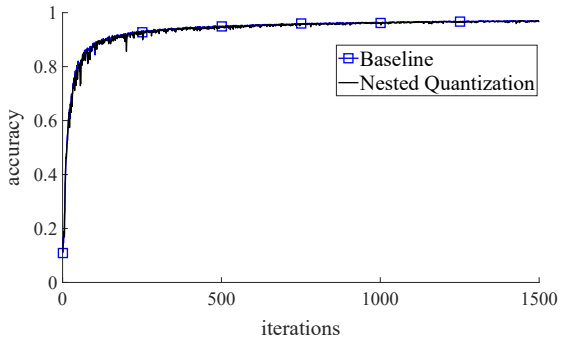
Table 2.1: Raw and compressed communication bits per worker (Mbits per iteration of training) for different networks. First row is the raw transmission rate and the second row is the quantized and compressed rate.

Method	Baseline	Nested Quantization	QSG	One-Bit
FC300-100	8.53	0.61	0.78	0.35
	—	0.328	0.36	0.339
Lenet	53.23	3.758	4.775	1.898
	—	2.42	2.673	1.895
CifarNet	34.19	2.435	3.088	1.254
	—	1.269	1.312	1.253

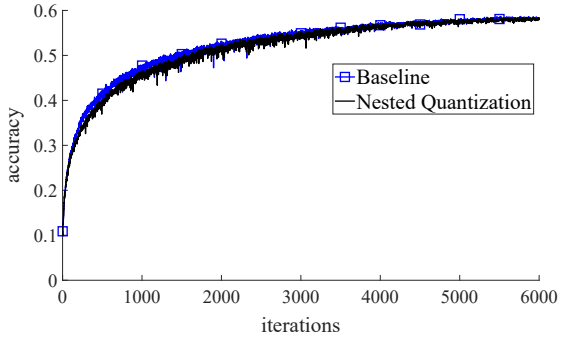
Next, we evaluate the performance of distributed training using distributed source coding. For this purpose, we considered FC-300-100 model and used SGD as the training algorithm. Our proposed scheme is based on dithered quantization of the stochastic gradients followed by distributed source coding using DISCUS (denoted as DQSG+DSC). Table 2.2 compares the proposed distributed coding (DQSG+DSC) with Nested Quantization, QSG [15] and one-bit [9] methods. Finally, figure 2.4 shows the convergence rate of the proposed scheme for 4 workers. It is worth noting that although the DQSG+DSC can further reduce the communication bit rate compared to the Nested quantization and QSG at the expense of



(a) Fully connected, 16 workers

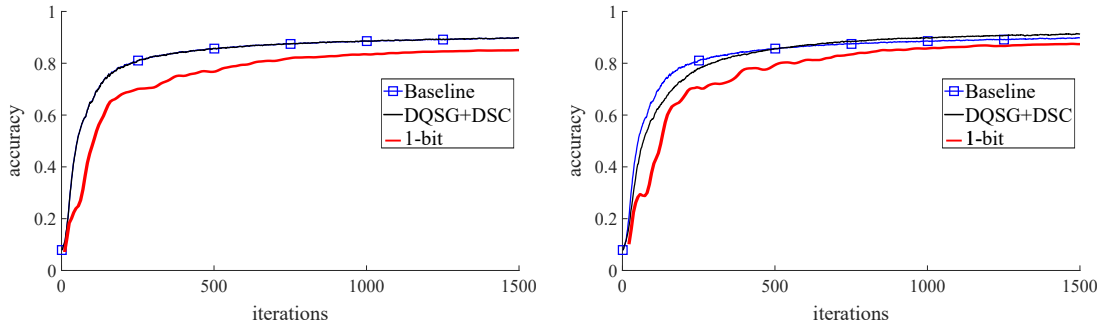


(b) Lenet, 16 workers



(c) CifarNet, 4 workers

Figure 2.3: Convergence rate of the distributed training using nested quantization compared to the baseline.



(a) 4 workers (b) 8 workers

Figure 2.4: Convergence rate of DQSG+DSC compared to the baseline for FC

more complex decoding at the server, its convergence speed is almost the same as the baseline (raw transmission). We have observed similar behavior for higher number of workers $(8, 16, \ldots)$.

Table 2.2: Communication bits per worker (Kbits per iteration of training) for different methods.

Workers	DQSG+DSC	Nested Q.	QSG	One-Bit
4	299.9181	374.3954	410.9895	340.8525
8	293.7495	354.5950	389.1265	340.0748
16	285.2091	328.2543	359.6669	339.1252

2.5 Conclusion

In this chapter, we argued that centralized distributed deep learning can be considered as a CEO problem; at each round of training the workers compute a noisy version of the true update (or stochastic gradient), and the goal is efficient transmission of the locally computed values to a central server to estimate the true update reliably. As such, we proposed a compression and estimation scheme, consisting of i) dithered and nested quantization at the workers, ii) distributed source coding to take advantage of the correlation among workers, and iii) decoding the data received from the workers and estimating the optimum parameters at the server. We showed that this approach can reduce the communication bit-rate, or alternatively, increase the precision of the aggregated SG at the server with similar or

less communication bit-rate. In our experiments, distributed learning with the CEO-based communication achieved it can achieve nearly the same convergence speed as of the baseline training.

CHAPTER 3

INDIRECT STOCHASTIC GRADIENT QUANTIZATION

3.1 Introduction

In the previous chapter, we have explored utilizing correlation among workers to reduce the communication bit-rate. However, the amount of correlation relies on the homogeneity of training data across workers and batch-size, which limits the its application and the amount of achievable compression gain. Moreover, generally, the existing communication methods that rely on directly compressing the stochastic gradients have either limited compression gains, high variance, or suffer from scalability issues as the total transmission bits scale almost linearly with the number of workers.

In this chapter, we consider compressing the SG matrix by factorizing it into low-rank matrices and then compressing them. However, naively pursuing such an approach is costly in distributed machine learning, in terms of the computational complexity and the training error. To overcome these issues, first, we take a deeper look at how the stochastic gradients are computed in practice. We observe that the cost function of a neural network w.r.t. the parameters of a layer, $\boldsymbol{W}$, can be reformulated as $\mathbb{E}_{\boldsymbol{x}}[f(\boldsymbol{W}\boldsymbol{x})]$ where $\boldsymbol{x}$ is the 'virtual' input of that layer. Therefore, we first consider the SG compression for this class of functions and develop a new algorithm, *indirect stochastic gradient quantization via factorization* (ISGQ). Then, we extend the algorithm to distributed training of deep neural networks. By analyzing the signals propagating in the neural networks, we observe that the forward and backward signals in neural networks are more compression-friendly than the stochastic gradients, themselves. Hence, ISGQ can achieve superior performance in terms of total transmission bits and quantization error compared to the traditional approaches.

3.2 Problem Statement and Motivation

To develop indirect stochastic gradient quantization, first we consider distributed learning of a generalized linear function given as $\mathcal{J}(\boldsymbol{W}) = \mathbb{E}_{\boldsymbol{x}}[f(\boldsymbol{W}\boldsymbol{x})]$.

For an arbitrary $\boldsymbol{x} \in \mathcal{X}$, $\boldsymbol{G} = \nabla_{\boldsymbol{W}} f(\boldsymbol{W}\boldsymbol{x}) = \nabla_{\boldsymbol{y}} f(\boldsymbol{y})|_{\boldsymbol{y}=\boldsymbol{W}\boldsymbol{x}}\, \boldsymbol{x}^{\mathsf{T}}$ is a SG of $\mathcal{J}$. It is common to compute and average the SG over a batch of data to reduce its variance. Let $\boldsymbol{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_L] \in \mathbb{R}^{n\times L}$ be a training batch of size L, $\boldsymbol{\delta}_k = \nabla_{\boldsymbol{y}} f(\boldsymbol{y})|_{\boldsymbol{y}=\boldsymbol{W}\boldsymbol{x}_k}$ for $k = 1, \ldots, L$ and $\boldsymbol{\Delta} = [\boldsymbol{\delta}_1, \cdots, \boldsymbol{\delta}_L] \in \mathbb{R}^{m\times L}$. Therefore,

$$\boldsymbol{G} = \frac{1}{L}\sum_{k=1}^{L} \boldsymbol{G}_k = \frac{1}{L}\sum_{k=1}^{L} \boldsymbol{\delta}_k \boldsymbol{x}_k^{\mathsf{T}} = \frac{1}{L}\boldsymbol{\Delta}\boldsymbol{X}^{\mathsf{T}}. \tag{3.1}$$

Our proposed method for quantization and compression of the stochastic gradients, computed via (3.1), is motivated by the following observation:
Instead of computing the gradients and then compressing them, our idea aims at compressing the intermediate signals, $\boldsymbol{\Delta}$ and $\boldsymbol{X}$, and transmitting them. We refer to this approach as indirect *compression, in contrast to the* direct *quantization and compression of the stochastic gradients $\boldsymbol{G}$. This is specially helpful when the number of parameters is large relative to the batch size; since the dimension of SG is $m \times n$, direct method requires transmission of mn values for $\boldsymbol{G}$. On the other hand, the indirect method requires transmitting only $L(m+n)$ values for a batch of size L. Moreover, as it will be investigated later, these signals are more compression-friendly, i.e., they tend to be sparser and having less entropy than the stochastic gradients.*

3.3 Indirect SG Quantization via Factorization

Here, we introduce and analyze the proposed indirect quantization of SG. Let $\widetilde{\boldsymbol{X}}$ and $\widetilde{\boldsymbol{\Delta}}$ be the quantized values of $\boldsymbol{X}$ and $\boldsymbol{\Delta}$, respectively. Then the indirect SG quantization (ISGQ) is

defined as

$$\widetilde{\boldsymbol{G}} = \frac{1}{L}\widetilde{\boldsymbol{\Delta}}\,\widetilde{\boldsymbol{X}}^{\mathsf{T}}. \tag{3.2}$$

Here, we focus on unbiased indirect quantizers, i.e., $\mathbb{E}\left[\boldsymbol{G} - \frac{1}{L}\widetilde{\boldsymbol{\Delta}}\,\widetilde{\boldsymbol{X}}^{\mathsf{T}}\right] = \mathbf{0}$. We consider two classes of quantizers for $\boldsymbol{X}$ and $\boldsymbol{\Delta}$, namely, *deterministic* and *random dithered* quantization.

3.3.1 Deterministic Indirect SG Quantization

We call a quantizer $Q(\cdot)$ *deterministic* if for any v, repeated application of the quantizer to v results in the same value. A quantizer $Q(\cdot)$ is statistically optimized for random variable z if it is unbiased and has the minimum mean squared error (MSE) [96, 97], hence [1]

$$\mathbb{E}_z[z - Q(z)] = 0, \quad \mathbb{E}_z[(Q(z) - z)\,Q(z)] = 0. \tag{3.3}$$

Let $g = G_{i,j}$ be an arbitrary element of the SG, $\boldsymbol{x} := (\boldsymbol{X}_{j,.})^{\mathsf{T}}$ and $\boldsymbol{\delta} := (\boldsymbol{\Delta}_{i,.})^{\mathsf{T}}$ be the j-th and i-th row of $\boldsymbol{X}$ and $\boldsymbol{\Delta}$, respectively. Hence, $g = \frac{1}{L}\boldsymbol{\delta}^{\mathsf{T}}\boldsymbol{x} = \frac{1}{L}\sum_k x_k\,\delta_k$. Further, assume that the signals have bounded joint second moment, i.e., $\mathbb{E}[\|\boldsymbol{x}\|^2\|\boldsymbol{\delta}\|^2] < \infty$.

One may hope that if the quantizers for $\boldsymbol{x}$ and $\boldsymbol{\delta}$ are designed optimally w.r.t. each individual signal, then the resulting indirect quantization of SG becomes almost optimal as well. We refer to this quantization approach as *naïve ISGQ*.

Lemma 3. *Assume that the quantizers $\boldsymbol{x}$ and $\boldsymbol{\delta}$ are designed optimally and $\tilde{g}$ is the naïve indirect quantization of g.*

– *If $\boldsymbol{x}$ and $\boldsymbol{\delta}$ are independent random variables, then $\tilde{g}$ is an unbiased and bounded-variance SG. Moreover, in 1-bit quantization, if x_k's are i.i.d. Folded Normal[2] and δ_k's are Normal random variables, then the MSE gap with the optimum direct quantizer is less than 4%.*

[1] Obviously, designing such a quantizer requires knowledge about the probability distribution of data or accessing the entire dataset.

[2] If U has Normal distribution $U \sim \mathcal{N}(0,1)$, then $V = |U|$ has folded normal distribution, denoted by $V \sim \mathcal{FN}(0,1)$.

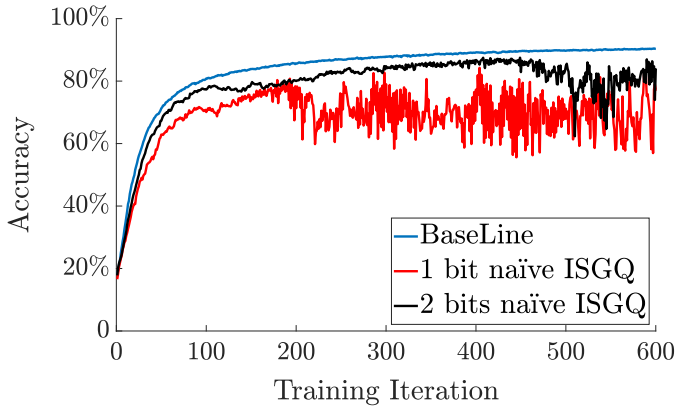


Figure 3.1: Performance of naïve-ISGQ w.r.t. the baseline (non-quantized) for training a fully connected deep model over MNIST.

– *If $\boldsymbol{x}$ and $\boldsymbol{\delta}$ are correlated random variables, the naïve ISGQ is not necessarily unbiased.*

Unfortunately, designing optimum individual quantizers for $\boldsymbol{x}$ and $\boldsymbol{\delta}$ is not feasible in many applications. Further, the independence assumption between $\boldsymbol{x}$ and $\boldsymbol{\delta}$ is not generally satisfied in practice and by Lemma 3, the naïve ISGQ is likely to become biased. These shortcomings limit the effectiveness of naïve ISGQ in many applications such as distributed deep learning (see Fig. 3.1).

The drawbacks of naïve-ISGQ are mainly due to the fact that the quantizers for the signals are designed independently, i.e., the quantized signals $\tilde{x}$ and $\tilde{\delta}$ are obtained by minimizing $\mathbb{E}[(x-\tilde{x})^2]$ and $\mathbb{E}\left[(\delta-\tilde{\delta})^2\right]$ separately without considering their joint effect on the computed SG. To overcome the problems of naïve ISGQ, we propose jointly optimizing the individual quantizers for $\boldsymbol{X}$ and $\boldsymbol{\Delta}$ such that the MSE of the resulting ISGQ is minimized. If the joint statistical properties of $\boldsymbol{X}$ and $\boldsymbol{\Delta}$ are available, one can aim at analytically finding optimum individual quantizers for unbiased minimum MSE ISGQ (please refer to the supplementary document). Here, we focus on empirical methods (using data of each training mini-batch) to approximately find good indirect quantizers.

Note that the quantization of $\boldsymbol{X}$ can be written as $\widetilde{\boldsymbol{X}} = \sum_{k=1}^{K} \boldsymbol{A}_k \alpha_k$, where K is the number of quantization bins, $[\boldsymbol{A}_k]_{i,j} = 1$ if $[\boldsymbol{X}]_{i,j}$ is in the k-th quantization bin (and $[\boldsymbol{A}_k]_{i,j} = 0$, otherwise) and α_k is the reconstruction point associated with the k-th bin.

Algorithm 1 Empirical MSE-ISGQ

1: Initialize $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$
2: **for** few iterations **do**
3: Fix $\boldsymbol{\alpha}$ and solve (3.6) to update $\boldsymbol{\beta}$.
4: Fix $\boldsymbol{\beta}$ and solve (3.6) to update $\boldsymbol{\alpha}$.
5: **return** Quantizers for $\boldsymbol{X}$ and $\boldsymbol{\Delta}$.

Similarly, we can represent quantization of $\boldsymbol{\Delta}$ as $\widetilde{\boldsymbol{\Delta}} = \sum_k \boldsymbol{B}_k \beta_k$. Therefore, ISGQ can be computed as

$$\widetilde{\boldsymbol{G}} = \frac{1}{L}\widetilde{\boldsymbol{\Delta}}\widetilde{\boldsymbol{X}}^{\mathsf{T}} = \frac{1}{L}\sum_{k,l} \boldsymbol{B}_l \boldsymbol{A}_k^{\mathsf{T}}\, \alpha_k \beta_l = \sum_{k,l} \boldsymbol{C}_{k,l}\, \alpha_k \beta_l. \tag{3.4}$$

where $\boldsymbol{C}_{k,l} = \frac{1}{L}\boldsymbol{B}_l \boldsymbol{A}_k^{\mathsf{T}}$. Define the empirical bias as

$$\begin{aligned} \text{bias} &:= \sum_{i,j} \big(G_{i,j} - \frac{1}{L}[\widetilde{\boldsymbol{\Delta}}\widetilde{\boldsymbol{X}}^{\mathsf{T}}]_{i,j}\big) \\ &= \sum_{i,j} \big[\boldsymbol{G} - \sum_{k,l} \boldsymbol{C}_{k,l}\alpha_k\beta_l\big]_{i,j} = \bar{G} - \boldsymbol{\beta}^{\mathsf{T}}\boldsymbol{P}\boldsymbol{\alpha}, \end{aligned} \tag{3.5}$$

where $\bar{G} = \sum_{i,j} G_{i,j}$ and $P_{k,l} = \sum_{i,j}[\boldsymbol{C}_{k,l}]_{i,j}$. Since the problem of optimizing the quantization bins for ISGQ is non-convex and computationally complex, we decide to fix them and only adjust the reconstruction points of each quantizer. Hence, the mappings $\boldsymbol{X} \mapsto \boldsymbol{A}_k$ and $\boldsymbol{\Delta} \mapsto \boldsymbol{B}_k$ are known. For example, in 1-bit ISGQ for correlated normal $\boldsymbol{X}$ and $\boldsymbol{\Delta}$, the quantization threshold is set to zero and only the reconstruction values for positive and negative $\boldsymbol{X}$ and $\boldsymbol{\Delta}$ are adjusted. We propose to adjust the quantizers for the empirical MSE-ISGQ via the optimization problem

$$\begin{aligned} &\min_{\boldsymbol{\alpha},\boldsymbol{\beta}} \ \|\boldsymbol{G} - \tilde{\boldsymbol{G}}\|_F^2 + \lambda(\text{bias})^2 \\ =&\min_{\boldsymbol{\alpha},\boldsymbol{\beta}} \ \|\boldsymbol{G} - \sum_{k,l} \boldsymbol{C}_{k,l}\alpha_k\beta_l\|_F^2 + \lambda(\boldsymbol{\beta}^{\mathsf{T}}\boldsymbol{P}\boldsymbol{\alpha} - \bar{G})^2, \end{aligned} \tag{3.6}$$

where λ controls the trade-off between the MSE and empirical bias of MSE-ISGQ.

Computational Complexity.

Since, the optimization problem (3.6) is bi-convex, we suggest the iterative approach summarized in Alg. 1 to solve it. The quantizers for $\boldsymbol{X}$ and $\boldsymbol{\Delta}$ can be initialized approximately based on the expected properties of the signals or as uniform quantizer.

Note that

$$\|\boldsymbol{G}-\widetilde{\boldsymbol{G}}\|_F^2 = \|\boldsymbol{G}-\sum_{k,l}\boldsymbol{C}_{k,l}\beta_l\alpha_k\|_F^2 =$$
$$\|\boldsymbol{G}\|_F^2+\sum_{k,l}\left(\alpha_k\alpha_l\sum_{i,j}\beta_i\beta_j\,\mathrm{trace}\big(\boldsymbol{C}_{k,i}\boldsymbol{C}_{l,j}^{\mathsf{T}}\big)\right)-2\sum_{k}\left(\alpha_k\sum_{i}\beta_i\,\mathrm{trace}\big(\boldsymbol{C}_{k,i}\boldsymbol{G}^{\mathsf{T}}\big)\right).$$

Define $[\boldsymbol{D}^{(k,l)}]_{i,j}=\mathrm{trace}\big(\boldsymbol{C}_{k,i}\boldsymbol{C}_{l,j}^{\mathsf{T}}\big)$ and $[\boldsymbol{E}]_{k,i}=\mathrm{trace}\big(\boldsymbol{C}_{k,i}\boldsymbol{G}^{\mathsf{T}}\big)$. Note that for K-level quantization, the total computational complexity and memory requirement of $\boldsymbol{D}$'s and $\boldsymbol{E}$ are $\mathcal{O}(K^4)$ which is negligible for small K. Moreover, these computations are only done once prior to optimizing $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. To analyze the complexity of each iteration of the optimization algorithm, let $\boldsymbol{\beta}$ be fixed. Set $[\boldsymbol{Q}]_{k,l}=\boldsymbol{\beta}^{\mathsf{T}}\boldsymbol{D}^{(k,l)}\boldsymbol{\beta}$, $\boldsymbol{r}=\boldsymbol{E}\boldsymbol{\beta}$ and $\boldsymbol{p}=\boldsymbol{P}^{\mathsf{T}}\boldsymbol{\beta}$. Then, the optimum $\boldsymbol{\alpha}$ can be found as

$$\underset{\boldsymbol{\alpha}}{\mathrm{argmin}}\left[\boldsymbol{\alpha}^{\mathsf{T}}\boldsymbol{Q}\boldsymbol{\alpha}-2\boldsymbol{r}^{\mathsf{T}}\boldsymbol{\alpha}+\lambda(\boldsymbol{p}^{\mathsf{T}}\boldsymbol{\alpha}-\bar{G})^2\right]=\left(\boldsymbol{Q}+\lambda\boldsymbol{p}\boldsymbol{p}^{\mathsf{T}}\right)^{-1}\left(\boldsymbol{r}+\lambda\bar{G}\boldsymbol{p}\right). \tag{3.7}$$

Fixing $\boldsymbol{\alpha}$ and optimizing $\boldsymbol{\beta}$ can be done similarly. In our experiments, we found out that only 1-2 iterations of Alg. 1 yields satisfactory results.

3.3.2 Dithered Indirect SG Quantization

The main drawback of using the deterministic approach for the quantization is the dependency of the quantization noise to the signal. Since $\boldsymbol{x}$ and $\boldsymbol{\delta}$ are generally correlated, this forced us in §3.3.1 to adjust the individual quantizers for each batch of data ($\boldsymbol{X}$ and $\boldsymbol{\Delta}$) to minimize the MSE and bias of ISGQ. The extra computational complexity due to the re-

quired optimization in MSE-ISGQ can adversely affect the training time in distributed deep learning. Here, we pursue a different approach and develop a *simple and fixed* quantization scheme whose noise is independent of the signals.

We consider the dithered indirect quantization of SG as follows: Let K_x and K_d be the number of desired quantization levels for $\boldsymbol{X}$ and $\boldsymbol{\Delta}$, respectively. $\boldsymbol{X}$ is quantized as

$$\boldsymbol{Q}_x = \lfloor \boldsymbol{X}/\kappa_x + \boldsymbol{U} \rceil\,, \quad \widetilde{\boldsymbol{X}} = \kappa_x \left(\boldsymbol{Q}_x - \boldsymbol{U}\right), \tag{3.8}$$

where the *scale factor* $\kappa_x = \|\boldsymbol{X}\|_\infty / K_x$ maps the signal into the range $[-K_x, K_x]$ prior to quantization and $\boldsymbol{U} \sim \mathcal{U}(-1/2, 1/2)$ is an independently generated random dither signal. It can be easily verified that the scaled quantization noise $\boldsymbol{E}_x = (\boldsymbol{X} - \widetilde{\boldsymbol{X}})/\kappa_x$ is independent of the signals $\boldsymbol{X}$ and $\boldsymbol{\Delta}$, and uniformly distributed over $(-1/2, 1/2)$. The dithered quantization of $\boldsymbol{\Delta}$ is defined similarly.

Theorem 4. *Let $\boldsymbol{G} = \frac{1}{L}\boldsymbol{\Delta}\,\boldsymbol{X}^\mathsf{T}$ be a stochastic gradient of $\mathcal{J}(\boldsymbol{W})$. Then, the Dithered-ISGQ, $\widetilde{\boldsymbol{G}} = \frac{1}{L}\widetilde{\boldsymbol{\Delta}}\widetilde{\boldsymbol{X}}^\mathsf{T}$ with number of quantization levels K_x and K_d, for $\boldsymbol{X}$ and $\boldsymbol{\Delta}$ respectively, has the following properties:*

P1. *$\widetilde{\boldsymbol{G}}$ is unbiased, i.e., $\mathbb{E}[\widetilde{\boldsymbol{G}}] = \nabla\mathcal{J}$,*

P2. *Its variance is bounded as $\mathbb{E}[\|\widetilde{\boldsymbol{G}} - \nabla\mathcal{J}\|_F^2] \le \frac{mn}{L}\gamma\,\mathbb{E}[\|\boldsymbol{X}\|_\infty^2\,\|\boldsymbol{\Delta}\|_\infty^2] + \mathbb{E}[\|\boldsymbol{G} - \nabla\mathcal{J}\|_F^2]$, where γ is a constant depending only on the number of quantization levels K_x and K_d.*

As a simple example, assume that $\boldsymbol{X}$ follows a Normal or Folded-Normal distribution with variance σ_x^2, and $\boldsymbol{\Delta} \sim \mathcal{N}(0, \sigma_d^2)$, where $\boldsymbol{X}$ and $\boldsymbol{\Delta}$ are generated independently. In this case, indirect quantization of $\boldsymbol{G} = \frac{1}{L}\boldsymbol{\Delta}\boldsymbol{X}^\mathsf{T}$ results in the MSE

$$\mathbb{E}\left[\|\widetilde{\boldsymbol{G}} - \nabla_{\boldsymbol{W}}\mathcal{J}\|_F^2\right] \le \frac{mn}{L}\sigma_x^2\sigma_d^2\left(\frac{\ln(\sqrt{2}nL)}{3K_x^2} + 1\right) \times \left(\frac{\ln(\sqrt{2}mL)}{3K_d^2} + 1\right).$$

Note that although the Dithered-ISGQ may have higher variance than MSE-ISGQ in some applications, it has the advantages of having fixed quantizers and not requiring

joint-optimization of the individual factorized quantizer.

Rate-Distortion Analysis.

It is worth exploring the relation between the variance of Dithered-ISGQ (i.e., the distortion) and the total number of bits (i.e., the transmission rate). Since the quantizer index of $\boldsymbol{X}$, $\boldsymbol{Q}_x \in \{-K_x, \dots, K_x\}$ can take at most $2K_x + 1$ distinct values and $\boldsymbol{X}$ has nL elements, the *total* number of bits for quantized $\boldsymbol{X}$ would be $nL \log(2K_x + 1)$. Similarly, total number of bits for quantized $\boldsymbol{\Delta}$ would be $mL \log(2K_d + 1)$. Hence, the total number of bits for dithered ISGQ is $R = L\big(n \log(2K_x + 1) + m \log(2K_d + 1)\big)$ per training iteration.

Considering the rate-distortion with respect to the batch-size L, we realize that $R = \mathcal{O}(L)$ while from (3.9) MSE$= \mathcal{O}(\frac{\ln(L)^2}{L})$. Thus, the rate increases linearly w.r.t. the batch-size but the decrease in quantization noise is sublinear.

Similarly, to analyze the rate-distortion w.r.t. the number of quantization levels, we observe that doubling the number of quantization levels increases the number of bits by 1 per sample. For sufficiently large nL and mL (relative to K_x and K_d), the MSE would be reduced approximately by a factor of 16. However, when $\ln(nL) \ll K_x^2$ and $\ln(mL) \ll K_d^2$, which corresponds to more quantization levels (i.e. finer quantization of $\boldsymbol{X}$ and $\boldsymbol{\Delta}$), the MSE of Dithered-ISGQ would be the same as the non-quantized SG and any further increase in the number of bits would not improve the accuracy anymore.

Computational Complexity.

It is worth mentioning that only the intermediate signals, $\boldsymbol{X}$ and $\boldsymbol{\Delta}$, are required to be available for Dithered-ISGQ and there is no need to compute the SG via (3.1). Moreover, quantizing $\boldsymbol{X}$ can be done in parallel while performing the forward and backward computations. Hence, generally the computation time of Dithered-ISGQ is less than other direct quantization methods.

Convergence Analysis.

The convergence analysis of the Dithered-ISGQ relies on the fact that the proposed quantization method is unbiased and has bounded variance. Consider stochastic gradient descent learning algorithm with ISGQ in which at the t-th iteration, the parameters are updated as

$$\boldsymbol{W}_{t+1} = \boldsymbol{W}_t - \eta_t \widetilde{\boldsymbol{G}}_t, \tag{3.9}$$

where η_t is the learning rate. Convergence of the learning algorithm can be easily verified under almost the same assumptions as in [98, §5.1], i.e.,

A1. $f(\cdot)$ is lower bounded and 3-times differentiable with continuous derivatives.

A2. Learning rates satisfy $\sum \eta_t = +\infty$ and $\sum \eta_t^2 < \infty$.

A3. Over the support of cost function $f(\cdot)$, the signals have bounded joint fourth moment $\mathbb{E}[\|\boldsymbol{X}\|_F^4 . \|\boldsymbol{\Delta}\|_F^4] < \infty$.

A4. If $\boldsymbol{W}$ grows too large, the gradient descent direction points towards zero.

Theorem 5. *Assume that conditions (A1) to (A4) hold. Then gradient descent with Dithered-ISGQ* (3.9) *converges almost surely to a local extremum, i.e.,* $\nabla_{\boldsymbol{W}_t} \mathcal{J} \overset{a.s.}{\to} 0$ *as* $t \to +\infty$.

3.4 Application to Distributed Training of Neural Networks

In this section, we show how ISGQ can be employed for efficient communication of stochastic gradients in distributed training of deep neural networks. Consider the l-th layer of a neural network, whose input signal is $\boldsymbol{x}^{(l-1)}$ and the weights and biases are $\boldsymbol{W}^{(l)}$ and $\boldsymbol{b}^{(l)}$, respectively. By concatenating $\boldsymbol{b}^{(l)}$ to $\boldsymbol{W}^{(l)}$ and appending 1 to $\boldsymbol{x}^{(l-1)}$ [3], the input signal

[3] i.e., $\boldsymbol{W}^{(l)} \leftarrow [\boldsymbol{W}^{(l)}, \boldsymbol{b}^{(l)}]$ and $\boldsymbol{x}^{(l)} \leftarrow [\boldsymbol{x}^{(l)}; 1]$.

into the nodes and the output of the l-th layer are given by

$$\boldsymbol{y}^{(l)} = \boldsymbol{W}^{(l)}\boldsymbol{x}^{(l-1)}\,, \quad \boldsymbol{x}^{(l)} = \boldsymbol{\sigma}(\boldsymbol{y}^{(l)}), \tag{3.10}$$

where $\boldsymbol{\sigma}(\cdot)$ is the activation function, applied element-wise.

There exists a function $g(\cdot)$ such that the final output of the neural network, $\boldsymbol{y}$, can be represented as $\boldsymbol{y} = g(\boldsymbol{x}^{(l)})$, where $g(\cdot)$ may depend on other signals and parameters of the neural network. Hence the loss function w.r.t. $\boldsymbol{x}^{(l)}$ and desired output $\boldsymbol{t}$ is given by $\ell(g(\boldsymbol{x}^{(l)}), \boldsymbol{t})$. By defining $f(\boldsymbol{v}) = \ell(g(\boldsymbol{\sigma}(\boldsymbol{v})), \boldsymbol{t})$, the training loss function with respect to the parameters of the l-th layer would be $\mathcal{J} = \mathbb{E}\big[f(\boldsymbol{W}^{(l)}x^{(l-1)})\big]$, where $x^{(l-1)}$ can be considered as the *virtual input* of the l-th layer.

Moreover, it is worth mentioning that the backpropagation algorithm, widely used in deep learning [99, 100], is indeed a realization of (3.1) and chain-rule. It is well-known that gradient of the cost function for an input $\boldsymbol{x}$ w.r.t. the parameters of the l-th layer can be computed as

$$\nabla_{\boldsymbol{W}^{(l)}}\mathcal{J} = \boldsymbol{\delta}^{(l)}\big(\boldsymbol{x}^{(l-1)}\big)^{\mathsf{T}}, \tag{3.11}$$

$$\boldsymbol{\delta}^{(l)} = \boldsymbol{\sigma}'(\boldsymbol{y}^{(l)}) \odot \big((\boldsymbol{W}^{(l+1)})^{\mathsf{T}}\boldsymbol{\delta}^{(l+1)}\big). \tag{3.12}$$

where $\delta_j^{(l)} := \frac{\partial f}{\partial y_j^{(l)}}$ is the partial derivative of the cost function w.r.t. input signal of the j-th node of the l-th layer, i.e., $\boldsymbol{\delta}^{(l)} = \nabla_{\boldsymbol{y}} f(\boldsymbol{y})|_{\boldsymbol{y}=\boldsymbol{W}^{(l)}\boldsymbol{x}^{(l-1)}}$. These observations imply the potential application of the ISGQ algorithms developed in §3.3 for the compression of SG and distributed training of deep models. Using ISGQ in distributed learning can provide the following benefits:

- Since calculating SGs at the workers is generally done via backpropagation algorithm, computing forward and backward signals does not incur extra computational complexity. On the other hand, in Dithered-ISGQ, there is no need to compute the SG via (3.11) and having access to $\boldsymbol{X}$ and $\boldsymbol{\Delta}$ (computed via (3.12)) is sufficient. Since

the complexity of quantizing individual signals is less than matrix multiplication, we argue that Dithered-ISGQ can slightly reduce the computational load at the workers in addition to reducing the total transmission bits.

- As the majority of signals are sparse due to the structure of neural networks and the forward and backward signals have generally less entropy, they are more compressible than the gradients (please refer to the supplementary document and [Anonymized]). For example, with ReLU activation function, $\sigma(y) = \sigma'(y) = 0$ for $y < 0$. Hence, the forward and backward signals ($\boldsymbol{x}$, $\boldsymbol{\delta}$) in the hidden layers are mostly sparse, and because of (3.10) and (3.12) their sparsities are correlated which can be used to further reduce the communication bit rate.

- Since the quantization of the signals are performed separately, it can be potentially implemented in parallel, and some operations (such as generating random dither signal) can be executed simultaneous to the neural network's forward and backward computations.

Note that the proposed indirect quantization is more suitable when the batch-size is smaller then the number of parameters. For layers with weight sharing schemes such as convolutional layers which generally have fewer parameters for transmission, distributed training benefits more from direct compression and transmission of the stochastic gradients using methods such as [15, 16].

3.5 Experiments

In this section, we evaluate the properties of the developed ISGQ algorithms and their performance in distributed training. For the simulations, we consider MNIST database with fully-connected (784-1000-300-100-10) neural network (hereafter referred to as FC) and Lenet model [94], CIFAR-10 database using CifarNet [95], and Imagenet [101] using AlexNet deep model [95]. The considered deep models, FC, Lenet, CifarNet and AlexNet

have approximately 1.16, 1.66, 1.07 and 62.4 million parameters, respectively. In all our experiments, we use stochastic gradient descent or Adam algorithm with batch-sizes 256 or 128 per worker. To evaluate the reduction in the transmission bits as well as the performance loss of the trained model, we compared our proposed method against the baseline distributed training without any quantization (i.e., 32 bits used for the transmissions of values) and other direct quantization methods: 1-bit quantization of [9], TernGrad [16] and QSGD [15]. For implementation details and the distributed learning algorithm, please refer to the supplementary document.

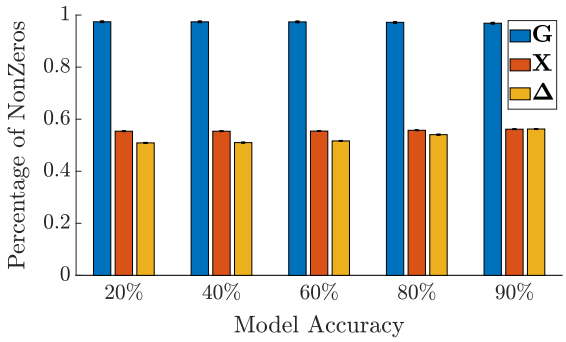
Quantizer Performance.

First, we investigate how our proposed ISGQ methods are compared against the direct Lloyd-Max quantization [97, 96][4]. For this purpose, we consider different neural networks at various stages of training and repeated the experiments numerous times to compute the mean and variance of the desired metrics. Some of the results are presented in figures 3.2 and 3.3.

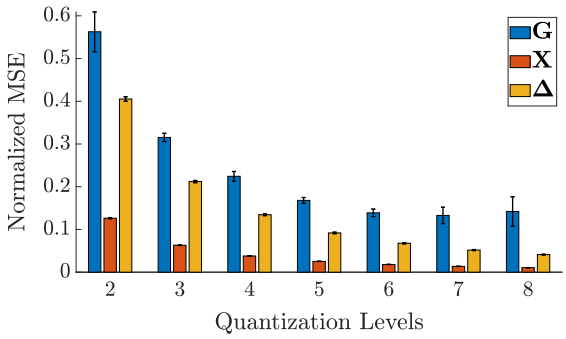
We observe that generally the forward and backward signals are sparser (Fig. 3.2a), and their optimum quantized values have less entropy and normalized MSE (defined as $\|\boldsymbol{v}-\tilde{\boldsymbol{v}}\|^2/\|\boldsymbol{v}\|^2$ for vector $\boldsymbol{v}$) than the SG (Fig. 3.2). Hence, quantization of the intermediate signals generally requires fewer number of bits and has smaller individual quantization noise than directly quantizing the signals, confirming that these signals are more compression-friendly.

Moreover, our proposed MSE-ISGQ (using only 1 iteration of Alg. 1) and Dithered-ISGQ usually performs comparable or better than the optimum (Lloyd-Max) direct quantization of the SG (see Fig. 3.3), showing the effectiveness of ISGQ.

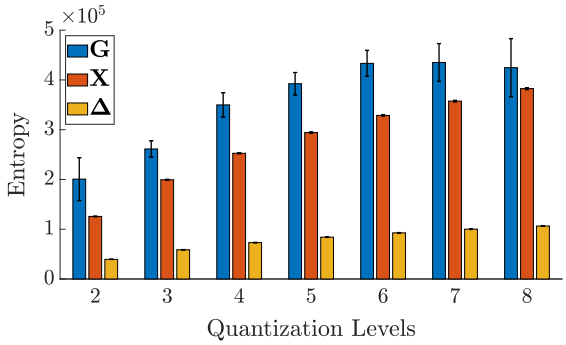
[4]The designed quantizer achieves lower MSE than other direct quantization techniques such as QSG, TernGrad and 1-bit.



(a) Percentage of Non-Zeros



(b) Normalized MSE



(c) Entropy

Figure 3.2: Sparsity at different stages of training, Normalized MSE and Entropy of quantized SG vs signals of the 2nd hidden layer of FC at accuracy=40% for various number of quantization levels.

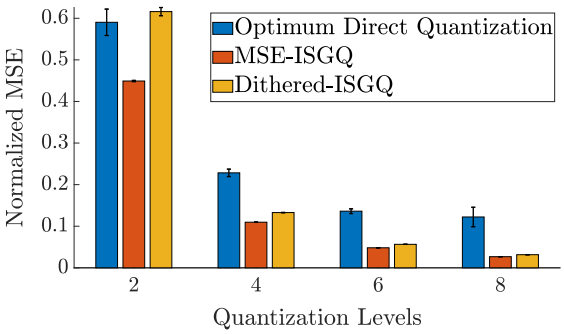


Figure 3.3: Comparing ISGQ with optimum direct SG quantization, second hidden layer of FC at accuracy=40%.

Processing Time per Iteration.

Next, we measure the complexity of the proposed SG compression technique by measuring the average time required to process (e.g., feed mini-batch and compute the SG), quantize and communicate the SGs. Let T_p be the total processing and quantization time and T_c be the average communication time to transmit the raw parameters of the model. Obviously, if a worker compresses the gradients by a factor of k, its communication time would be reduced approximately by Tc/k, while on the other hand, its processing time might increase slightly. As a result, in a centralized synchronous distributed training with P identical workers, the total processing time would be $T_p + PT_c/k + T_u$, where T_u is the communication time to broadcast back the aggregated gradients to the workers by the server.

First, we compare the required total processing and quantization times of the proposed ISGQ with QSG [15] and baseline (no quantization) for different batch-sizes and different models using Intel Core i7 CPU and Nvidia Titan Xp GPU. Since, baseline transmission only computes the SGs, the total processing time is expected to be larger when quantization is added. Tables 3.1 and 3.2 show the results for processing 200 batches on CPU and GPU, respectively. Since the dithered-ISGQ does not require computing the SG via (3.11) and only relies on back-propagation calculations, when matrix multiplications are costly (e.g., on CPU or for large matrices), its computation time is significantly lower than other quantization techniques and comparable to the baseline.

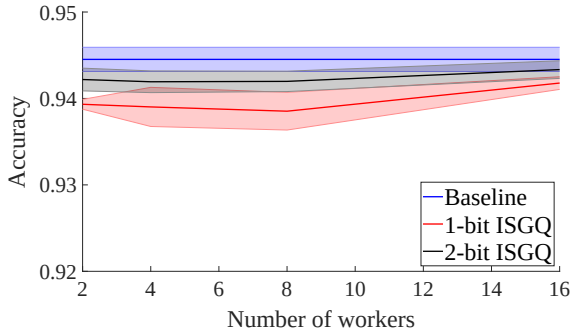


Figure 3.4: Final accuracy of the trained FC model, shaded areas represent 1 standard deviation.

Next, to find the effectiveness of different quantization schemes in terms of communication overhead, we calculated and compared compression gain of each scheme as

$$\text{compression gain} = \frac{32 \times (\text{\# model's parameters})}{\text{\# transmitted bits per worker}}.$$

Some of the results are presented in Tbl. 3.3 for different models, batch-sizes and various quantization schemes.

One can easily conclude that 1000 iterations of decentralized distributed training Alexnet with 4 workers, batch-size 128 per worker using Titan Xp GPUs connected via InfiniBand links would take approximately 3.9 minutes using ISGQ compared to 4.5 minutes by QSG and 9 minutes by Baseline (no SG compression), while centralized single node training with the same total batch-size takes approximately 14.8 minutes to execute.

Accuracy of the Distributed Training.

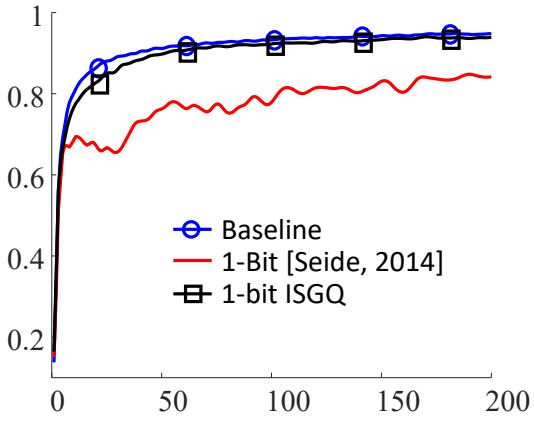
Although it is possible to evaluate the performance of the quantization and compression schemes in both synchronous and asynchronous settings, here we assume that the workers and server are synchronous. The main reason for such a setting is to cancel-out the performance degradation (in terms of training accuracy or speed) that may be caused by the stale gradients in asynchronous updates and to solely compare the effect of the quantization algorithms. Through our simulations, we have found that distributed training of the considered

Table 3.1: Computation time (sec.) with Core i7 CPU

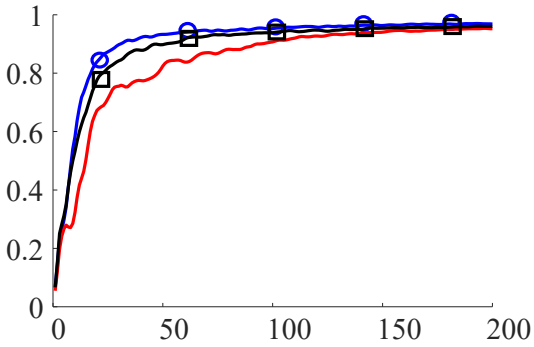
	Batch size	256	128	64
FC	Baseline	1.2	0.78	0.63
	QSGD	1.85	1.43	1.23
	D-ISGQ	1.29	0.76	0.52
Lenet	Baseline	14.4	8.17	5.1
	QSGD	15.79	9.1	5.9
	D-ISGQ	15.12	8.45	4.98
CifarNet	Baseline	30.33	16.31	9.2
	QSGD	31.59	17.1	9.92
	D-ISGQ	31.4	16.77	9.19
Alexnet	Baseline	66.4	34.5	18.9
	QSGD	70	37.6	21.8
	D-ISGQ	66.7	34.4	18.4

Table 3.2: Computation time (sec.) w/ Titan Xp GPU

	Batch size	256	128	64
FC	Baseline	0.29	0.26	0.25
	QSGD	0.34	0.32	0.31
	D-ISGQ	0.36	0.33	0.31
Lenet	Baseline	1.27	0.84	0.62
	QSGD	1.39	0.98	0.77
	D-ISGQ	1.41	1.0	0.79
CifarNet	Baseline	3.27	1.62	0.92
	QSGD	3.34	1.69	0.99
	D-ISGQ	3.26	1.7	1.01
Alexnet	Baseline	83	45.2	25
	QSGD	86	46.1	25.5
	D-ISGQ	84	44.4	24.1



(a) FC



(b) Lenet

Figure 3.5: Convergence rate of distributed training with 8 workers using different quantization methods.

Table 3.3: Average compression gains of different methods in distributed learning

	Batch size	256	128	64
FC	1-bit ISGQ	33	67	133
	1-bit quantization [9]	28.4	28.4	28.4
	TernGrad / 1-bit QSGD	20.2	20.2	20.2
Lenet	1-bit ISGQ	56	105	180
	1-bit quantization [9]	28	28	28
	TernGrad / 1-bit QSGD	20	20	20
CifarNet	1-bit ISGQ	38	65	98
	1-bit quantization [9]	28	28	28
	TernGrad / 1-bit QSGD	20.1	20.1	20.1
AlexNet	1-bit ISGQ	117	170	221
	2-bits ISGQ	80	118	153
	1-bit quantization [9]	29	29	29
	TernGrad / 1-bit QSGD	19.4	19.4	19.4

deep models using either of the quantization schemes eventually converges to $\pm 1\%$ of the accuracy of the baseline model. However, the convergence speed of the 1-bit method [9] is considerably slower than the others for complex models, while ISGQ performs comparably well. For example, Fig. 3.4 compares the final accuracy of the trained model with ISGQ using different number of workers with the baseline[5]. As seen, the accuracy loss due to the training with quantized SG is small (less than 0.2% most of the time for 2-bit ISGQ).

Figure 3.5 shows the test accuracy of the model at each iteration during training with stochastic gradient descent using baseline (no quantization), 1-bit quantization [9] and our proposed ISGQ. Note that here we omitted the time delays that is caused by more communication overhead in the baseline and 1-bit quantization and assumed that the speed of connection link is infinity. As shown in the figure, the convergence rate of ISGQ closely follows the baseline while it has the potential of achieving compression gains of beyond 32, much higher than the traditional direct quantization methods. On the other hand, the convergence rate of 1-bit quantization is severely affected by the larger quantization noise.

[5]We ran experiments multiple times with different initializations to find the average and standard deviation of the final trained model.

3.6 Conclusion

In this chapter, we proposed a novel approach, *indirect stochastic gradient quantization via factorization*, instead of commonly used direct methods. Our method takes advantage of the characteristics of the backpropagation algorithm and the statistical properties of the forward and backward signals during training. For the quantization of the forward and backward signals, we proposed two approaches; optimizing the quantization points such that the error in the reconstructed SG is minimized, and random dithered quantization of the factorized terms. We showed that despite its simplicity, ISGQ can perform close to the Lloyd-Max quantization algorithm in terms of the reconstruction error while requiring fewer bits. Moreover, ISGQ leads to significant reduction in the communication overhead, achieving compression gain of more than 100, without sacrificing the training speed or accuracy. Especially for a fixed total batch-size, at each worker the required transmission bit-rate of the fully connected layers decreases as the number of workers increases. This results in the reduction of total bits for transmission of the parameters in ISGQ, in contrast to the existing direct approaches whose transmission bit-rate remains fixed regardless of the number of workers.

CHAPTER 4
QUANTIZED COMPRESSIVE SAMPLING

4.1 Introduction

In chapters 2 and 3, we have proposed two different techniques based on the CEO problem and indirect SG quantization via factorization. However, the performance of the CEO-based communication is limited by the amount of the correlation among workers and the amount of compression achievable by ISGQ depends on the structure of neural network and batch-size. In this chapter, we investigate the problem of achieving arbitrarily large compression gains while ensuring that the compressed SGs are unbiased and have low variance.

The existing quantization methods have drawbacks such as

- Due to the quantization noise, the total variance of the SG would be increased. Hence, the learning algorithm with quantized SG may not converge with the same set of training hyper-parameters as the baseline algorithm. As a result, the hyper-parameters must be adjusted to ensure the convergence of the learning algorithms, which in turn can increase the required *number of training iterations* for the convergence of the model.

- If the quantizer is biased (e.g., sign SGD), the training algorithm is not guaranteed to converge (see, e.g., [28]).

- Since the small gradients are suppressed by the larger ones and thus would be most likely quantized to zero, the parameters whose gradients are relatively small may not be updated even if their gradients point to the same direction in multiple consecutive iterations of training.

Although using error-feedback [27, 28] can alleviate these issues to some extent, the

requirement to store the residual of quantization at each worker increases the memory footprint of the training algorithm significantly.

In this chapter, we introduce our proposed Quantized Compressive Sampling (QCS) method for the compression of stochastic gradients or parameter updates of deep models.

4.2 Quantized Compressive Sampling of Stochastic Gradient

Let $\boldsymbol{g} \in \mathbb{R}^n$ be the stochastic gradient of the model. Instead of directly compressing $\boldsymbol{g}$, our proposed method is based on mapping $\boldsymbol{g}$ onto $\mathbb{R}^k$, $k \leq n$, via $\boldsymbol{v} = \boldsymbol{T}\boldsymbol{g}$ and then compressing $\boldsymbol{v}$. Here, $\boldsymbol{T}$ is a random mixing matrix chosen from a class of appropriate transforms $\mathcal{T}$. Inspired by the work on structured measurement matrix in compressed sensing, we consider the following class of random mixing matrices

$$\boldsymbol{T} = \frac{1}{\sqrt{k}} \boldsymbol{H}\boldsymbol{R}, \tag{4.1}$$

where $\boldsymbol{R}$ is a random Rademacher diagonal matrix, i.e., $\boldsymbol{R} = \mathrm{diag}(\boldsymbol{r})$, $\mathrm{P}(r_i = 1) = \mathrm{P}(r_i = -1) = 0.5$, and $\boldsymbol{H}$ is constructed by picking up the first k rows[1] from the Hadamard matrix $\boldsymbol{H}_n \in \mathbb{R}^{n \times n}$. Note that the random transformation can be alternatively applied as

$$\boldsymbol{v} = \frac{1}{\sqrt{k}} \boldsymbol{H}(\boldsymbol{r} \odot \boldsymbol{g}). \tag{4.2}$$

Lemma 6. *The random mixing matrix $\boldsymbol{T} = \frac{1}{\sqrt{k}}\boldsymbol{H}\boldsymbol{R}$ has the following properties:*

$$\boldsymbol{T}\boldsymbol{T}^{\mathsf{T}} = \frac{n}{k}\boldsymbol{I}, \quad \mathbb{E}\left[\boldsymbol{T}^{\mathsf{T}}\boldsymbol{T}\right] = \boldsymbol{I}. \tag{4.3}$$

The quantization and compression of $\boldsymbol{v}$ is based on dithered quantization [91, 102]. Let

[1] It is possible to choose any arbitrary or random subset of k rows from $\boldsymbol{H}_n$, but the performance and analysis would be the same.

Q be the desired range of quantization levels and $\boldsymbol{u} \sim \mathcal{U}(-1/2, 1/2)$ be the random dither signal, independent of $\boldsymbol{v}$. The dithered quantization of $\boldsymbol{v}$ is computed as

$$\boldsymbol{q} = \lfloor \boldsymbol{v}/\varrho + \boldsymbol{u} \rceil \,, \tag{4.4}$$

where the *scale factor* $\varrho = \|\boldsymbol{v}\|_\infty / Q$ maps the elements of $\boldsymbol{v}$ into the range $[-Q, Q]$. For 1-bit dithered quantization, set $\varrho = 2\|\boldsymbol{v}\|_\infty$ and

$$\boldsymbol{q} = \operatorname{sign}(\boldsymbol{v}/\varrho + \boldsymbol{u}). \tag{4.5}$$

The *Quantized Compressive Sampling* (QCS) of $\boldsymbol{g}$ is then computed via first dequantizing $\boldsymbol{v}$ as

$$\widehat{\boldsymbol{v}} = \varrho \, (\boldsymbol{q} - \boldsymbol{u}) \,, \tag{4.6}$$

and then estimating $\boldsymbol{g}$ from $\widehat{\boldsymbol{v}}$. Note that the quantization of $\boldsymbol{v}$ can be written as

$$\widehat{\boldsymbol{v}} = \boldsymbol{v} + \varrho \boldsymbol{\varepsilon}, \tag{4.7}$$

where the scaled quantization noise $\boldsymbol{\varepsilon}$ is independent of the signals and $\boldsymbol{\varepsilon} \sim \mathcal{U}(-1/2, 1/2)$.[2] Note that although $\boldsymbol{T}$ is a random matrix, the server can reproduce it by using the same random number generators and seed numbers. We consider two different criteria for reconstructing $\boldsymbol{g}$; 1) minimizing the mean squared error $\mathbb{E}[\|\boldsymbol{g} - \widehat{\boldsymbol{g}}\|_2^2]$ and 2) finding an unbiased estimator. To have simple yet efficient estimation of $\boldsymbol{g}$ from $\widehat{\boldsymbol{v}}$, we restrict ourselves to the class of linear estimators given by

$$\widehat{\boldsymbol{g}} = \boldsymbol{A}^{\mathsf{T}} \widehat{\boldsymbol{v}}, \tag{4.8}$$

where $\boldsymbol{A} = \alpha \boldsymbol{T}$ and α is a scalar which may depend on ϱ but is independent of $\boldsymbol{g}$.

[2]Note that this is not the case for ordinary quantization or stochastic quantization (QSG) of [15].

4.2.1 Unbiased Estimator

We constraint the reconstruction matrix such that the resulting quantizer be unbiased, $\mathbb{E}[\widehat{\boldsymbol{g}}] = \boldsymbol{g}$, for any arbitrary $\boldsymbol{g}$. Using Lemma 6, it can be easily verified that for an *unbiased QCS*, the reconstruction matrix is given by

$$\alpha = 1, \quad \boldsymbol{A} = \boldsymbol{T}. \tag{4.9}$$

The following theorem summarizes the properties of the proposed QCS.

Theorem 7. *The QCS with $\alpha = 1$ is unbiased and has bounded variance error. More specifically, for an arbitrary $\boldsymbol{g} \in \mathbb{R}^n$, let $\widehat{\boldsymbol{g}} = \boldsymbol{T}^{\mathsf{T}}\widehat{\boldsymbol{v}}$ be the QCS of $\boldsymbol{g}$ and $\boldsymbol{e} = \boldsymbol{g} - \widehat{\boldsymbol{g}}$. Then,*

P1. *The quantizer is unbiased, i.e., $\mathbb{E}[\boldsymbol{e}] = \boldsymbol{0}$.*

P2. *The variance of error is bounded as $\mathbb{E}[\|\boldsymbol{e}\|_2^2] \leq \gamma \|\boldsymbol{g}\|_2^2$ where γ is a constant given by*

$$\gamma = \begin{cases} \frac{n}{k} - 1 + \frac{n}{4Q^2}\,\frac{\log(k)}{k-1} & k \geq 2 \\ n - 1 & k = 1 \end{cases} \tag{4.10}$$

Thm. 7 provides a trade-off between the number of transmission bits per value and the variance of QCS. Assuming that the overhead to transmit scale factor ϱ is negligible, the total transmission bits would be $k \log(2Q+1)$ and hence the compression gain is

$$\text{gain} = \frac{nb}{k \log_2(2Q+1)}, \tag{4.11}$$

where b is the number of bits used in representing each parameter (generally, in floating point computations $b = 32$). For a fixed compression gain, minimizing (4.10) would result in the optimum number of quantization levels Q and k. Figure 4.1 shows the minimum achievable γ using the proposed unbiased QCS and compares it with QSG (Lemma 3.1 of [15]) and the

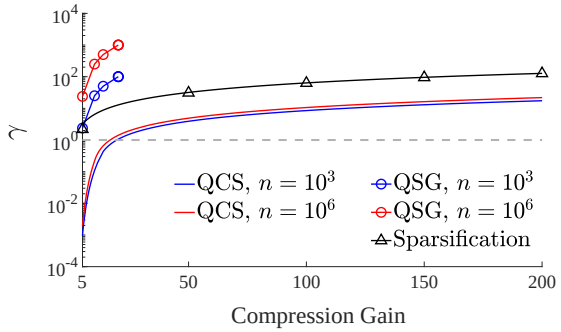


Figure 4.1: Variance bound γ vs. compression gain.

lower bound of the expected compression gain of the unbiased sparsification [25]. Note that the compression gain of [15] is at most 32. As shown in the figure, the variance bound of our proposed unbiased QCS is orders of magnitude lower than both other approaches.

4.2.2 Minimum Mean Squared Error Estimator

In *MMSE-QCS*, the objective is finding the reconstruction matrix such that $\mathbb{E}[\|\widehat{\boldsymbol{g}} - \boldsymbol{g}\|_2^2]$ is minimized. However the quantizer is not necessarily unbiased. In this case, the reconstruction matrix is approximately given by setting

$$\alpha = \frac{1}{\gamma + 1}, \tag{4.12}$$

where γ is as in (4.10).

Lemma 8. *For an arbitrary $\boldsymbol{g} \in \mathbb{R}^n$, let $\widehat{\boldsymbol{g}} = \alpha \boldsymbol{T}^{\mathsf{T}} \widehat{\boldsymbol{v}}$ be the QCS of $\boldsymbol{g}$ and $\boldsymbol{e} = \boldsymbol{g} - \widehat{\boldsymbol{g}}$. Then, for α given by* (4.12)*, we have*

$$\mathbb{E}\left[\|\boldsymbol{e}\|_2^2\right] \leq (1 - \alpha)\|\boldsymbol{g}\|_2^2. \tag{4.13}$$

Algorithm 2 summarizes the proposed quantization and reconstruction for Unbiased-QCS and MMSE-QCS. Note that both QUANTIZE and DEQUATNIZE functions generate the

Algorithm 2 Quantized Compressive Sampling of SG

function QUANTIZE($\boldsymbol{g}, \boldsymbol{H}, Q$)
  Generate random Rademacher vector $\boldsymbol{r}$.
  Generate random dither $\boldsymbol{u} \sim \mathcal{U}(-1/2, 1/2)$.
  $\boldsymbol{v} \leftarrow \frac{1}{\sqrt{k}} \boldsymbol{H}(\boldsymbol{r} \odot \boldsymbol{g})$
  $\varrho \leftarrow \|\boldsymbol{v}\|_\infty / Q$
  $\boldsymbol{q} \leftarrow \lfloor \boldsymbol{v}/\varrho + \boldsymbol{u} \rceil$
  return $\boldsymbol{q}$ and ϱ

function DEQUANTIZE($\boldsymbol{q}, \varrho, \boldsymbol{H}$)
  Set α. ▷ via (4.9) or (4.12)
  Reproduce random Rademacher vector $\boldsymbol{r}$.
  Reproduce random dither $\boldsymbol{u} \sim \mathcal{U}(-1/2, 1/2)$.
  $\widehat{\boldsymbol{v}} = \varrho\,(\boldsymbol{q} - \boldsymbol{u})$
  $\widehat{\boldsymbol{g}} = \frac{\alpha}{\sqrt{k}} \boldsymbol{r} \odot (\boldsymbol{H}^\mathsf{T} \widehat{\boldsymbol{v}})$
  return $\widehat{\boldsymbol{g}}$

same random Rademacher and uniform sequences via utilizing identical random number generation algorithms with the same seed values.

4.3 Weighted Error Feedback

Application of quantization or sparsification techniques in deep learning may introduce two major issues: (i) increase in the variance of the aggregated gradients, and (ii) insertion of a bias to the stochastic gradient. These may degrade the convergence speed or even cause the learning algorithm fail to converge. A key component in tackling both of these issues is aggregating the compression residuals (i.e., quantization or sparsification errors) and carrying forward to the next mini-batch. This ensures that the true values of SG are eventually applied to the parameters of the deep model, although it may take several transmissions, i.e., it resembles stale (partial) gradient updates. Exploiting such a feedback can speed up the convergence rate or ensure the convergence of the learning algorithms such as stochastic gradient descents even in the presence of (biased) gradient compression [19, 27, 28].

Since adding quantization error from previous steps can potentially increase the overall

variance of SG and the staleness of the gradients, we add a forgetting factor β in the error feedback which is a crucial part in bounding the variance of error feedback as we will show next. Let $\boldsymbol{r}_t$ be the running compression residue at the t-th iteration, with $\boldsymbol{r}_0 = \mathbf{0}$, and COMPRESS denotes quantizing and then dequantizing using Alg. 2. At the t-th iteration of training, we have

$$\boldsymbol{z}_t \leftarrow \boldsymbol{g}_t + \beta \boldsymbol{r}_t \tag{4.14a}$$

$$\widehat{\boldsymbol{z}}_t \leftarrow \text{COMPRESS}(\boldsymbol{z}_t) \tag{4.14b}$$

$$\boldsymbol{e}_t \leftarrow \boldsymbol{z}_t - \widehat{\boldsymbol{z}}_t \tag{4.14c}$$

$$\boldsymbol{r}_{t+1} \leftarrow (1-\beta)\boldsymbol{r}_t + \boldsymbol{e}_t \tag{4.14d}$$

and the parameters of the model are updated using $\widehat{\boldsymbol{z}}_t$ instead of SG $\boldsymbol{g}_t$. The next lemma states the sufficient conditions on β for the residual signal $\boldsymbol{r}_t$ be ℓ_2 bounded.

Lemma 9. *Assume that the SGs are ℓ_2 bounded, i.e., $\mathbb{E}[\|\boldsymbol{g}\|_2^2] \leq B$. Then, $\mathbb{E}[\|\boldsymbol{r}_t\|_2^2] \leq \eta B$, where*

- *for Unbiased-QCS,*

$$\eta = \frac{\gamma}{1 - ((1-\beta)^2 + \beta^2\gamma)}, \tag{4.15}$$

  for $0 < \beta < \min(1, 2/(1+\gamma))$.

- *For MMSE-QCS,*

$$\eta = \frac{\gamma}{\left(\sqrt{\gamma + 1 - (1-\beta)^2} - \sqrt{\gamma}\right)^2}, \tag{4.16}$$

  for $0 < \beta \leq 1$

Note that for Unbiased-QCS, since γ might be greater than 1, the residual signal's magnitude may become unbounded for $\beta = 1$ (i.e., the traditional error feedback method), and hence the learning algorithm would not converge with error feedback. Note that for Unbiased-QCS, since γ might be greater than 1, $\beta = 1$ (traditional error feedback) can

cause the residual signal's magnitude to blow up and prevent the convergence of the learning algorithm using error feedback. On the other hand, in MMSE-QCS all values of $0 \leq \beta \leq 1$ are viable choices for convergence with the error feedback.

Remark 3. We can choose β to minimize the upper bound on the ℓ_2 norm of the residual signal. In this case, the optimum values of β for Unbiased-QCS and MMSE-QCS are given by (4.17) and (4.18), respectively;

$$\beta_u^* = \frac{1}{\gamma+1}, \quad \eta_u^* = \gamma(\gamma+1) \tag{4.17}$$

$$\beta_m^* = 1, \qquad \eta_m^* = \frac{\gamma}{\left(\sqrt{\gamma+1}-\sqrt{\gamma}\right)^2}. \tag{4.18}$$

Moreover, as it can be easily verified, $\eta_u^* < \eta_m^*$. Hence, the *theoretical* upper bound for Unbiased-QCS with weighted error feedback is smaller than MMSE-QCS.

Remark 4. Using Lemma 3 of [28], by simple derivations and noting that δ in their notation is the same as $1/(\gamma+1)$ for MMSE-QCS, we realize that the upper bound in [28] equals to $\eta_k = 4\gamma(\gamma+1)$ which can be easily verified that it is larger than η_m^* derived here.

4.4 Convergence Analysis

In this section, we show the convergence of the proposed SG compression algorithms with and without error feedback. In our analysis, we consider the gradient descent algorithm with the compressed stochastic gradients and we make the following assumptions;

Assumption 1. *The loss function is Lipschitz-smooth, i.e., there exists a constant L such that for all $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$*

$$\|\nabla \mathcal{J}(\boldsymbol{w}_1) - \nabla \mathcal{J}(\boldsymbol{w}_2)\|_2 \leq L\|\boldsymbol{w}_1 - \boldsymbol{w}_2\|_2. \tag{4.19}$$

Assumption 2. *The stochastic gradients are ℓ_2 bounded, i.e., $\exists B > 0$ such that*

$$\mathbb{E}\left[\|\boldsymbol{g}\|_2^2\right] \leq B. \tag{4.20}$$

Remark 5. Note that Assumption 2 can be relaxed to have bounded variance SG, i.e., $\mathbb{E}[\|\boldsymbol{g} - \nabla f\|_2^2] \leq \sigma^2$ for some constant σ. The analysis would be slightly more involved, however the convergence results would be similar to the ones that are stated here. (see supplementary document)

First, we consider training for T iterations of SGD with fixed step-size μ using Unbiased-QCS and MMSE-QCS without any error feedback, i.e., at the t-th iteration, the parameters are updated as

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \mu \widehat{\boldsymbol{g}}_t, \tag{4.21}$$

where $\widehat{\boldsymbol{g}}_t$ is compressed SG from either Unbiased-QCS or MMSE-QCS.

Lemma 10. *Let f^* be the minimum of objective function $f(\cdot)$. Assuming* (4.19) *and* (4.20) *hold, in training with Unbiased-QCS, we get*

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\left[\|\nabla f(\boldsymbol{w}_t)\|_2^2\right] \leq \frac{f(\boldsymbol{w}_0) - f^*}{T\mu} + \frac{L}{2}\mu(1+\gamma)B.$$

Similarly, for MMSE-QCS we have

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\left[\|\nabla f(\boldsymbol{w}_t)\|_2^2\right] \leq (1+\gamma)\frac{f(\boldsymbol{w}_0) - f^*}{T\mu} + \frac{L}{2}\mu B.$$

In both cases, by appropriate choice of step size, we can achieve $\mathcal{O}(1/\sqrt{T})$ convergence rate

$$\min_t \mathbb{E}\left[\|\nabla f(\boldsymbol{w}_t)\|_2^2\right] \leq \frac{f(\boldsymbol{w}_0) - f^* + \frac{L}{2}(1+\gamma)B}{\sqrt{T}}. \tag{4.22}$$

Comparing the convergence rates of Unbiased-QCS and MMSE-QCS with that of the SGD with uncompressed gradients, we observe that both achieve asymptotically the same rate of convergence $\mathcal{O}(1/\sqrt{T})$, however the constant term in the rate is slightly larger due to the compression.

Next, we consider the effect of using weighted error feedback on the convergence of the training algorithm. At the t-th iteration, the parameters are updated as

$$\begin{aligned}
\boldsymbol{z}_t &\leftarrow \boldsymbol{g}_t + \beta \boldsymbol{r}_t \\
\widehat{\boldsymbol{z}}_t &\leftarrow \textsc{Compress}(\boldsymbol{z}_t) \\
\boldsymbol{e}_t &\leftarrow \boldsymbol{z}_t - \widehat{\boldsymbol{z}}_t \\
\boldsymbol{w}_{t+1} &\leftarrow \boldsymbol{w}_t - \mu \widehat{\boldsymbol{z}}_t \\
\boldsymbol{r}_{t+1} &\leftarrow (1-\beta)\boldsymbol{r}_t + \boldsymbol{e}_t
\end{aligned}$$

The following lemma proves the convergence of the training algorithm.

Lemma 11. *Let f^* be the minimum of objective function $f(\cdot)$ and assume* (4.19) *and* (4.20) *hold. Then,*

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\big[\|\nabla f(\boldsymbol{w}_t)\|_2^2\big] \leq \frac{f(\boldsymbol{w}_0) - f^*}{T\mu/2} + LB\big(\mu + 4L\eta\mu^2\big),$$

where η is given by (4.15) *for Unbiased-QCS and by* (4.16) *for MMSE-QCS.*

With a slightly tighter analysis and setting $\mu = \frac{\sqrt{1+\epsilon}}{\sqrt{T}}$ for arbitrary $\epsilon > 0$, we have

$$\min_t \mathbb{E}\big[\|\nabla f(\boldsymbol{w}_t)\|_2^2\big] \leq \frac{f(\boldsymbol{w}_0) - f^* + \frac{L}{2}B(1+\epsilon)}{\sqrt{T}} + L^2 B \frac{(1+\epsilon)^2}{\sqrt{1+\epsilon}-1}\frac{\eta}{T}. \tag{4.23}$$

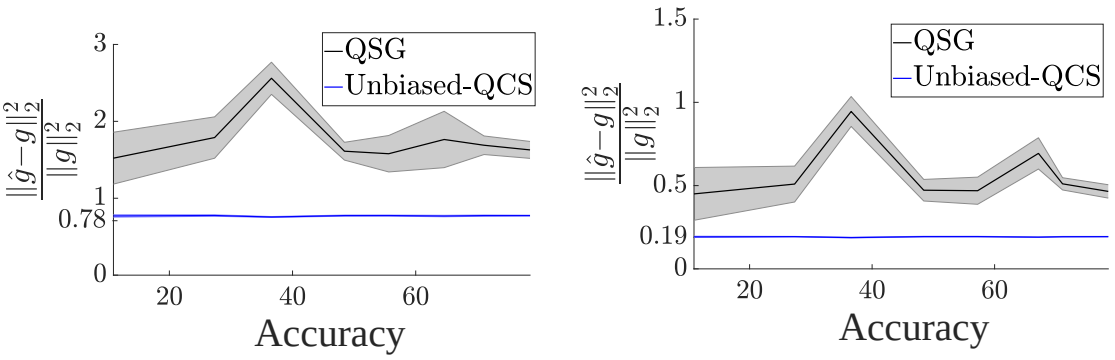
Comparing the convergence rates of (4.22) and (4.23) with that of SGD, we observe that the excess term in the convergence rate due to the compression of SG are proportional to $\gamma/\sqrt{T}$ and η/T, respectively, for training without and with feedback. When $\gamma \ll 1$, $\eta \approx \gamma$

and using error feedback dwarfs the effect of the compression on the convergence by an additional factor $1/\sqrt{T}$. On the other hand, for high compression gains and hence large γ, we have $\eta \approx \gamma^2$. Using error feedback reduces the term in (4.23) due to the compression of SG from $\mathcal{O}(1/\sqrt{T})$ to $\mathcal{O}(1/T)$, resulting in faster diminishing of the extra term and closing the gap with the SGD.

4.5 Experiments and Discussions

Our experiments are divided into three parts. First, we evaluate the performance of the proposed quantization methods. Next, we investigate the execution time of training with the proposed quantizers and finally, we evaluate the performance of distributed learning using different number of workers and various quantization parameters. To evaluate our algorithms, we considered a fully connected neural network with hidden layers of sizes $1000 - 300 - 100$ (herein, referred to as FC) and a Lenet-5 like convolutional network [94] over MNIST, a convolutional network on Cifar10 (referred to as CifarNet) and Alexnet [95] over Imagenet database. We compare QCS-SG with various communication bit-rates against the baseline (no quantization of gradients), 1-bit quantization [9], QSG [15] and Top-K SGD [25]. In most cases, the experiments were repeated 10-100 times to obtain reliable results for mean and variance of the behavior of the desired quantities.

In our implementation of QCS, we divided the gradients into partition to reduce the complexity of the algorithm and improve its performance, similar to the approach suggested in [15]. Depending on the size of each layer's parameters, the partition sizes were chosen to be a power of 2 or from the set $\{96, 100, 192, 200, 288, 320, 384\}$. For these choices, the Hadamard matrices are designed using Sylvester's, Payley's or Williamson's construction algorithm.



(a) compression gain = 21

(b) compression gain = 14

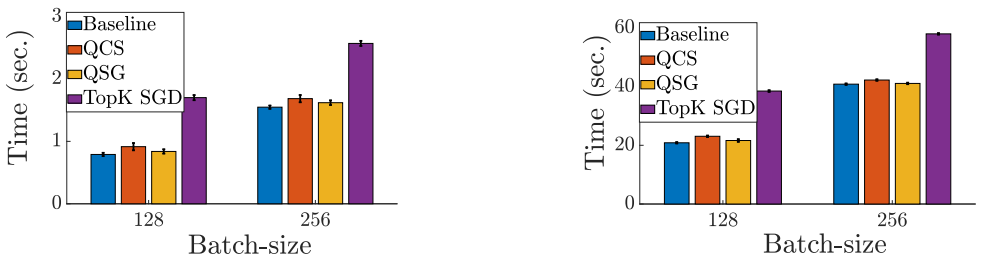
Figure 4.2: Relative quantization error vs. accuracy of model during training of Lenet over MNIST. Shaded areas represent 1σ variations.

Quantizer Evaluation.

To examine the effectiveness of the quantization scheme, we measured the relative quantization error, defined as $\frac{\|\boldsymbol{g}-\widetilde{\boldsymbol{g}}\|_2^2}{\|\boldsymbol{g}\|_2^2}$, for different models, datasets and with different number of quantization levels. Figure 4.2 compares the relative quantization error of Unbiased-QCS against QSG [15] during training for different models and compression gains. The results confirm our findings in Thm. 7 and theoretical comparisons in 4.1. It is worth noting that unlike QSG, the relative quantization error of QCS is highly concentrated around the mean value. This suggest that *training with QCS-SG is similar to training with unquantized SG corrupted by a (Gaussian) noise with fixed signal to noise ratio.*.

Processing Time.

We measured the required time to compute and quantize the gradients for processing 100 batches of training data using different batch-sizes (not accounting for loading data from HDD or communicating among workers) and compared with the baseline (no quantization) and QSG [15] over a Titan Xp GPU. Figure 4.3 shows the results for different batch-sizes per worker. We note that although the compression gain of our proposed QCS can become arbitrarily large, its processing time is slightly higher that QSG and much lower than Top-K



(a) Cifarnet (b) Alexnet

Figure 4.3: Time to process and compress SG for 100 batches

sparsification.

As an example, 100 iterations of decentralized distributed training Alexnet with 4 workers, batchsize 128 per worker using Titan Xp GPUs connected via InfiniBand links would take approximately 22 seconds using QCS with compression gain 100, compared to 27 seconds by QSG, 42 seconds by Top-K SGD[3], and 55 seconds by Baseline (no SG compression), while centralized single node training with the same total batch-size takes approximately 90 seconds to execute.

It is worth noting that as the models become more complex and the number of parameters increases, the overhead of applying transforms to the partitions of SG, which have small size $d < 500$, becomes negligible relative to the computational complexity of the backpropagation algorithm. Hence, the more desirable properties of QCS and its relatively negligible overhead compared to QSG and other quantization methods make QCS a favorable choice for distributed learning of large deep models.

Performance in Distributed Deep Learning.

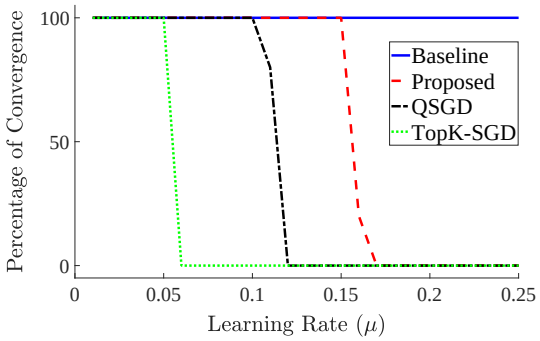
To analyze the convergence rate of QCS, first we evaluate the effect of compression noise on training a simple linear regression problem, and next we consider distributed deep learning.

Linear Regression- Consider learning a linear regression model $\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x}$ with mean squared error (MSE) cost function $\mathcal{J} = 0.5\,\mathbb{E}[\|\boldsymbol{y} - \boldsymbol{W}\boldsymbol{x}\|_2^2]$, where $\boldsymbol{y} \in \mathbb{R}^m$ is the desired

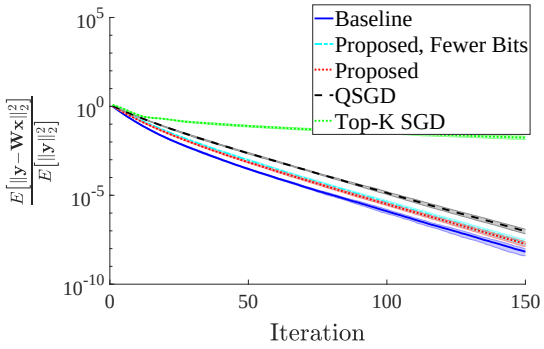
[3]The parameters are chosen to achieve the same compression gain.

(target) signal and $\boldsymbol{x} \in \mathbb{R}^n$ is the input. Assume that $\boldsymbol{x}$ is a zero-mean multivariate Gaussian random vector with correlation matrix $\boldsymbol{R}$ whose maximum and minimum eigenvalues are $\lambda_{\max}(\boldsymbol{R}) = 4$ and $\lambda_{\min}(\boldsymbol{R}) = 1$, respectively. It is known that gradient descent with step-size $\mu < 1/\lambda_{\max} = 1/4$ converges to the optimal solution. To investigate the impact of compressing SG on the convergence rate, we consider learning $\boldsymbol{W}$ via stochastic gradient descent algorithm with batch-size 32 and using no quantization (baseline), QSGD [15], Top-K SGD [25], and our proposed method. The parameters are adjusted such that compression gains of all methods are approximately 21, except the method labeled as '*Proposed, Fewer Bits*' in Fig. 4.4b which uses approximately 40% fewer bits. We set $n = 64$, $m = 50$ and repeated the experiments several times with different values of learning rate to obtain the range of μ that the training algorithm converges and the corresponding convergence rate. Figure 4.4a shows the percentage of times different learning algorithms converge vs. step-size μ. We nose that quantization or sparsification reduces the range of μ for which SGD converges. However, our proposed method significantly increases that range compared to existing methods. Although using smaller μ ensures the convergence for QSGD and Sparsified SGD (see Fig. 4.4b), it sacrifices the potential of higher convergence rates that can be achieved by using larger step-sizes (Fig. 4.4c). In this example, our proposed method consistently outperforms the other existing algorithms. Even by step-size $\mu = 0.10$, the convergence time can be reduced by a factor of 2 compared to the QSGD with $\mu = 0.05$.

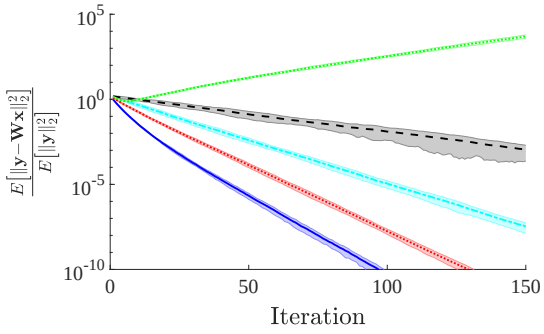
Distributed Deep Learning- We evaluate the convergence and the number of communication bits in a distributed learning system with different number of workers. In our simulations, the batch-size per worker is fixed at 128. Hence, by increasing the number of workers, the effective total batch-size increases. Although it is possible to evaluate the performance of the quantization and compression schemes in both synchronous and asynchronous settings, here we assume that the workers and server are synchronous. The main reason for such a setting is to cancel-out the performance degradation (in terms of training accuracy or speed) that may be caused by the stale gradients in asynchronous updates, and



(a) Convergence region for step-size μ

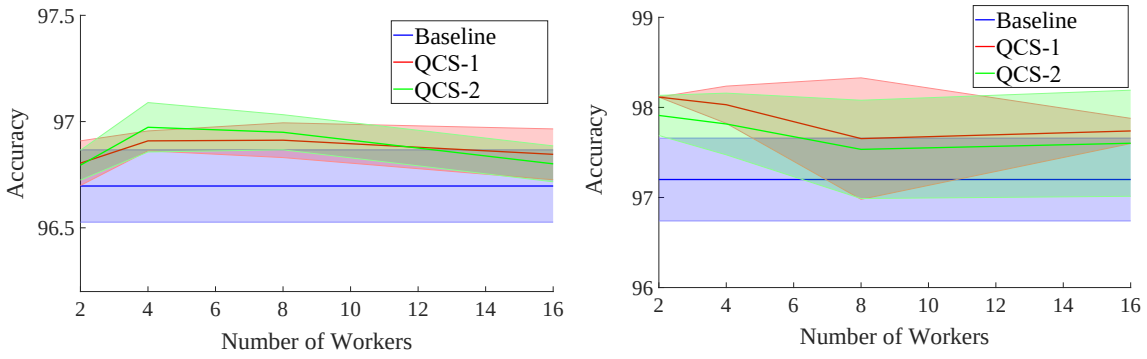


(b) Convergence rate, $\mu = 0.05$



(c) Convergence rate, $\mu = 0.1$

Figure 4.4: Effect of different compression techniques on the convergence of SGD on learning simple linear regression model. The shaded region represents variations of ± 1.5 standard deviation. Note that the scale of convergence plots is logarithmic.



(a) FC

(b) Lenet

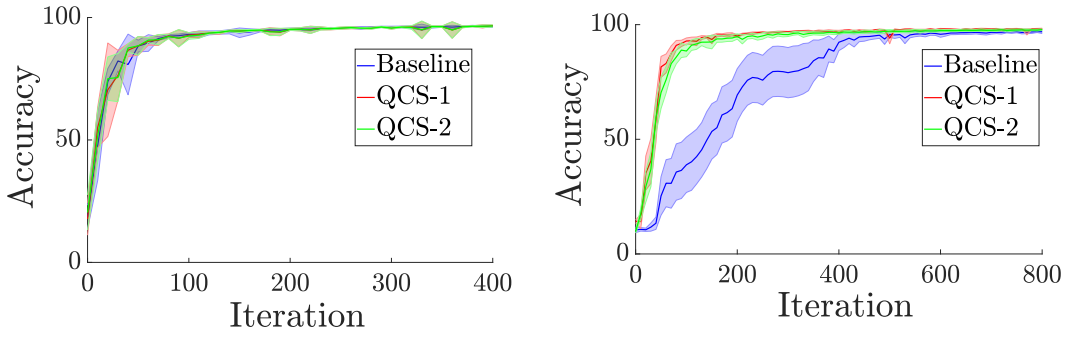
Figure 4.5: Accuracy of distributed training vs number of workers, using SGD learning algorithm

to solely investigate the effect of the quantization/compression algorithms.

We consider two different settings: QCS-1 achieves compression gain of approximately 32 by optimally setting k and Q (see Thm. 7 and the discussion after), and in QCS-2 $k = n$ and $Q = 1$. Hence, QCS-2 achieves the same compression gain as QSG. Figure 4.5 shows the accuracy of the final trained model vs different number of workers for FC and Lenet models. Moreover, in Figures 4.6 and 4.7, we have compared the convergence rate of QCS w.r.t. baseline (no quantization) for different settings. It is interesting to note that QCS improves the convergence rate of the training in some occasions compared to the baseline (no quantization). We believe this is mainly due to the characteristics of the quantization noise. Since the noise from the QCS behaves similar to a (Gaussian) noise with fixed signal to noise ratio, our method is likely to result in a better convergence property than the aforementioned techniques for complex training data [103, 104].

4.6 Conclusion

The performance of the CEO-based communication is affected by the amount of correlation among the workers. On the other hand, the compression gain and performance of ISGQ is limited by the structure of neural network and batch-size. In this chapter, we considered the problem of achieving an arbitrarily large *unbiased* compression of SG. To achieve that goal,



(a) FC

(b) Lenet

Figure 4.6: Convergence rate of distributed training of FC and Lenet models, 4 workers with SGD learning algorithm

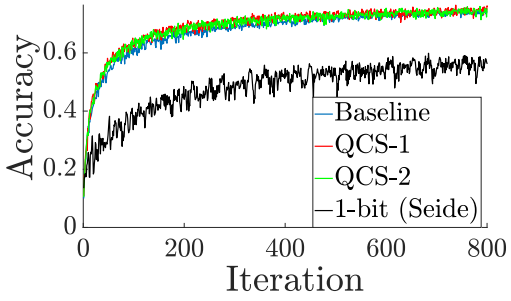


Figure 4.7: Convergence rate of distributed training of CifarNet, 4 workers with Adam learning algorithm

we considered projecting the SGs into a small subspace via a random linear transformation, and then quantize the signals in the lower dimension space. We showed that by using appropriate random transformation and dithered quantization, the proposed technique (QCS) can achieve orders of magnitude smaller MSE compared to the state-of-the-art unbiased compression techniques. For non-convex optimization problems, stochastic gradient descent with the proposed QCS compression enjoys the same convergence rate of $\mathcal{O}(1/\sqrt{T})$ as the baseline training, where T is the number of training iterations. However, since the constant factor in the rate is slightly larger due to the compression, there is an $\mathcal{O}(1/\sqrt{T})$ gap w.r.t. the baseline training. We showed that utilizing *weighted* error feedback reduces the effect of SG compression on convergence rate to $\mathcal{O}(1/T)$ [53]. The experiments confirm that QCS is computationally fast and has better convergence rate than the considered methods.

CHAPTER 5
FEDERATED LEARNING OVER WIRELESS MULTIPLE ACCESS CHANNELS

5.1 Introduction

The training data in a wireless edge network is generally unevenly distributed over a large number of nodes with limited resources such as communication bandwidth and battery power. Transferring data from edge nodes to a central server to train a deep model is often infeasible due to the limited wireless bandwidth and battery power as well as privacy concerns in some applications. Hence, it is desired to train the deep model over an edge network in a distributed manner. Federated learning [56, 54, 55, 59] enables such networks to collaboratively learn a unified deep model without transmitting the training data to a central server.

Federated learning differs from traditional distributed machine learning as 1) the number of edge nodes is generally very large, 2) the data observed by the nodes are usually unbalanced and non-iid, and 3) some nodes may not transmit at each round of communication. The majority of existing methods to reduce communication overhead in distributed learning rely on quantizing the SGs [9, 29, 15, 13, 10, 12, 50, 51, 53, 52], sparsification [17, 18, 24, 19, 25, 21] or a combination of both. However, direct application of these compression methods requires transmission of the compressed values without any interference from other nodes in the wireless channel. Therefore, such approaches require channel assignments to individual nodes (e.g., through TDMA or FDMA), which increases the latency. The majority of past works in federated learning over wireless Multiple Access Channel (MAC) are restricted to the transmission of raw (uncompressed) SGs or parameter updates [105, 106, 107, 108]. The exceptions are [109, 110] which implicitly require SGs to have almost the same sparsity patterns, and thus limiting their use to the iid datasets where the SGs

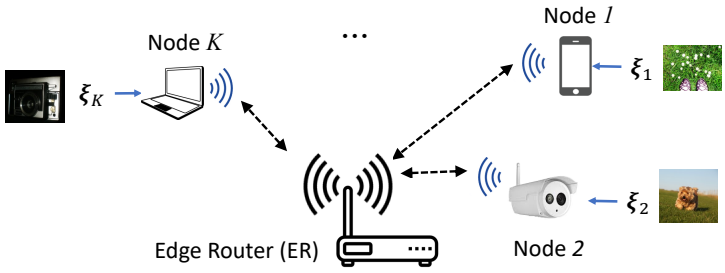


Figure 5.1: Wireless Edge Network

computed by the edge nodes have similar sparsity pattern. In contrast, we seek to develop a framework that incorporates the requirements of ML in wireless networks, and exploits properties such as over-the-air computation in wireless-MAC.

5.2 Problem Statement

Figure 5.1 illustrates the wireless edge network considered throughout this chapter. *We will refer to the edge device as edge node or simply a node throughout*. The uplink communication is over a wireless Multiple Access Channel (MAC), which naturally performs an analog over-the-air addition on incoming signals from the edge nodes to the router. However, the downlink communication from the edge router to the edge nodes is wireless broadcast. Like edge nodes, the edge router is also assumed to have some memory and computing power.

For the communication between edge nodes and the edge router (ER), we assume symbol level synchronization (e.g., via a synchronization channel or synchronized clocks). During the uplink transmission, let $\boldsymbol{x}_i \in \mathbb{R}^m$ be the symbols transmitted by the i-th node. The received signal at the ER is given by

$$\boldsymbol{y} = \sum_i \boldsymbol{h}_i \odot \boldsymbol{x}_i + \boldsymbol{\eta}, \tag{5.1}$$

where $\boldsymbol{h}_i \in \mathbb{C}^m$ is subchannels' gains from node i to ER, and $\boldsymbol{\eta} \sim \mathcal{CN}(0, \sigma^2 \boldsymbol{I})$ is the MAC

channel noise, assumed to be complex Gaussian and independent across subchannels. In the downlink, if ER broadcasts $\boldsymbol{y}$ to the edge network, each node receives a noisy scaled replica of $\boldsymbol{y}$. For simplicity, we assume the channel state information is available at the nodes. Hence, by compensating for the downlink channel gains, the reconstructed value at the i-th node is given by $\widehat{\boldsymbol{y}}_i = \boldsymbol{y} + \boldsymbol{\eta}_i'$, where $\boldsymbol{\eta}_i' \sim \mathcal{N}(\mathbf{0}, \sigma_i^2 \boldsymbol{I})$.

Consider training a deep model with a cost function $F(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\xi}}[\ell(\boldsymbol{\xi}; \boldsymbol{\theta})] \approx \frac{1}{n} \sum_{\boldsymbol{\xi} \in \mathcal{X}} \ell(\boldsymbol{\xi}; \boldsymbol{\theta})$, where $\boldsymbol{\theta} \in \mathbb{R}^d$ is the parameters of the deep model, $\ell(\boldsymbol{\xi}; \boldsymbol{\theta})$ is the loss function of the model corresponding to input data $\boldsymbol{\xi}$, $\mathcal{X}$ is the training dataset and $n = |\mathcal{X}|$ is the number of training samples. Assume that node i observes only subset $\mathcal{X}_i \subset \mathcal{X}$, $|\mathcal{X}_i| = n_i$. Hence, its local objective function is $f_i(\boldsymbol{\theta}) = \frac{1}{n_i} \sum_{\boldsymbol{\xi} \in \mathcal{X}_i} \ell(\boldsymbol{\xi}; \boldsymbol{\theta})$, and the total cost function can be reformulated as $F(\boldsymbol{\theta}) = \sum_i \alpha_i f_i(\boldsymbol{\theta})$, where $\alpha_i = {}^{n_i}/_{n}$ is introduced to compensate for unbalanced training data sets among edge nodes.

Here, we focus on federated learning over wireless edge, with the focus on compressing SGs to reduce communication overhead. Further, we require the compression algorithm to be tailored to satisfy the constraints imposed by the communication medium and take advantage of its characteristics, i.e.,

P1 The MAC channel (5.1) can naturally compute weighted average of the transmitted values.

P2 The transmission power of each individual node is bounded, i.e., $\mathbb{E}[\|\boldsymbol{x}_i\|^2] \leq P_i$.

P3 All edge nodes may not transmit at every round of communication.

P4 Edge-node's private information should not leak to ER.

5.3 Preliminaries

The high level diagram of federated learning over wireless MAC is shown in Fig. 5.2. Let $\boldsymbol{g}_i \in \mathbb{R}^d$ be the stochastic gradient computed at node i, such that $\mathbb{E}[\boldsymbol{g}_i] = \nabla f_i(\boldsymbol{\theta})$. Therefore,

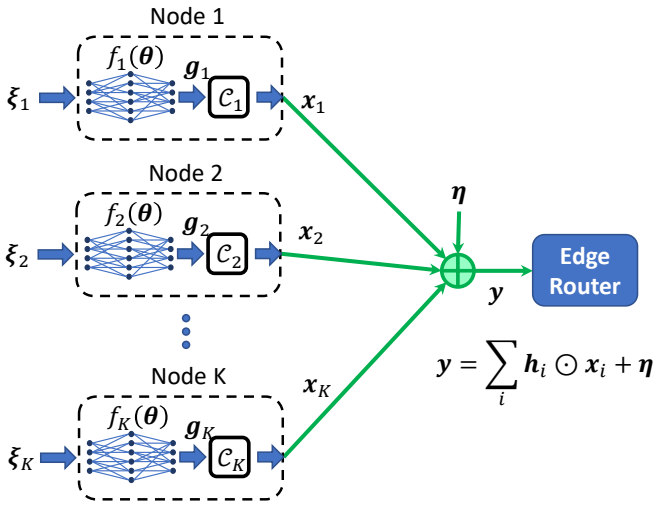


Figure 5.2: Federated learning over Wireless MAC. Node i observes data $\boldsymbol{\xi}_i$ and based on its local model, computes the model's stochastic gradient $\boldsymbol{g}_i$. Compression engine $\mathcal{C}_i$ compresses $\alpha_i \boldsymbol{g}_i$ to $\boldsymbol{x}_i$ and transmits over MAC. The edge router receives the noisy aggregated data $\boldsymbol{y} = \sum_i \boldsymbol{h}_i \odot \boldsymbol{x}_i + \boldsymbol{\eta}$ and broadcasts it back to the edge nodes.

the SG of $F(\boldsymbol{\theta})$ would be given as $\boldsymbol{g} = \sum_i \alpha_i \boldsymbol{g}_i$. For each node, our goal is to design an efficient encoding algorithm $\mathcal{C}_i(\cdot) : \mathbb{R}^d \to \mathbb{R}^m$ to compress scaled SGs, where $m \ll d$ and will be selected to control the trade offs among the wireless bandwidth requirement, the communication latency, and the training convergence rate.

For simplicity, we assume that the channel state information and hence $\boldsymbol{h}_i$ is known at node i. After compensating for the channel loss[1], $\boldsymbol{x}_i = \boldsymbol{h}_i^{-1} \odot \mathcal{C}_i(\alpha_i \boldsymbol{g}_i)$ would be the transmitted signal at node i. Then the received signal at the ER is given by

$$\boldsymbol{y} = \sum_{i \in \mathcal{K}} \mathcal{C}_i(\alpha_i \boldsymbol{g}_i) + \boldsymbol{\eta}, \tag{5.2}$$

where $\mathcal{K} \subset \{1, 2 \ldots, K\}$ is the subset of nodes transmitting their data. The aggregated signal $\boldsymbol{y}$ is then broadcasted back to the nodes to estimate an SG of $F(\boldsymbol{\theta})$. Ideally, at each node, we wish to be able to compute $\boldsymbol{g} = \sum_i \alpha_i \boldsymbol{g}_i$, i.e., the stochastic gradient of the objective function $F(\boldsymbol{\theta})$. However, due to the limited bandwidth, channel noise and the loss

[1]Note that here, for the presentation simplicity, we did not ignore sub-channels with huge losses. However, in practice, those poor channels can be discarded during data transmission.

of information due to the compression by $\mathcal{C}_i(\cdot)$'s, the estimated SG may not be the same as $\boldsymbol{g}$. We consider two additional criteria in developing the encoders $\mathcal{C}_i(\cdot)$'s:

C1 For privacy, given $\boldsymbol{y}$, the ER should not be able to infer any information about individual $\boldsymbol{g}_i$'s.

C2 Each participating node should be able to estimate an *unbiased* stochastic gradient of $F(\boldsymbol{\theta})$ from $\boldsymbol{y}$. This ensures the convergence of the SG-based learning algorithms. Otherwise, the training procedure can drift away from converging to the optimum (or good) solution, unless the bias in SG is compensated by error-feedback [28, 27, 53]. This, in turn, increases the memory footprint of the compression algorithm.

Note that the above conditions imply that there exists a function $\mathcal{D}(\cdot)$, such that $\widehat{\boldsymbol{g}} = \mathcal{D}(\boldsymbol{y})$ gives an unbiased estimation of $\boldsymbol{g}$, i.e., $\mathbb{E}[\mathcal{D}(\boldsymbol{y})] = \boldsymbol{g} = \sum_i \alpha_i \boldsymbol{g}_i$. As a measure of privacy, we require that the server should not be able to infer extra information about individual $\boldsymbol{g}_i$'s, as long as at least two nodes are transmitting their data. Specifically, if there is another set of stochastic gradients $\{\boldsymbol{g}_i'\}_i$ which results in the same SG for $F(\boldsymbol{\theta})$, they should not be distinguishable. This implies that for a given $\boldsymbol{g}$ and for all $\{\boldsymbol{g}_i\}_i$ such that $\sum_i \alpha_i \boldsymbol{g}_i = \boldsymbol{g}$, the output $\sum_i \mathcal{C}_i(\alpha_i \boldsymbol{g}_i)$ should not depend on the individual $\boldsymbol{g}_i$'s and is dependent only on $\boldsymbol{g}$.

Lemma 12. *The conditions in* **C1** *and* **C2** *impose a Homomorphic property on the encoder. As such, it is necessary that the encoders, $\mathcal{C}_i(\cdot)$'s, be identical linear transforms for all i.*

As a result of Lemma 12, we focus on the encoders given by $\mathcal{C}_i(\boldsymbol{z}) = \boldsymbol{A}\boldsymbol{z}$, where $\boldsymbol{A} \in \mathbb{R}^{m \times d}$ to be designed. On the other hand, note that if $\boldsymbol{A}$ is chosen to be fixed and deterministic, the information in the SGs residing in the Null space of $\boldsymbol{A}$ would be lost, hindering the learning algorithm from exploring the entire space of parameters while trying to minimize the objective function. As such, it is crucial to change $\boldsymbol{A}$ every few iterations of training to allow the SGs to navigate different directions in the parameter space.

One possible approach is generating elements of $\boldsymbol{A}$, a_{ij}, iid according to a zero-mean distribution such as Gaussian, Rademacher or $a_{ij} \in \{-1, 0, +1\}$. However, inspired by QCS

(chapter 4, [53]), in the proposed *Random Linear Coding*, we restrict $\boldsymbol{A}$ to be of the form $\boldsymbol{A} = \frac{1}{\sqrt{m}}\boldsymbol{H}\boldsymbol{R}$ where $\boldsymbol{H} \in \{\pm 1\}^{m\times d}$ is a partial Hadamard matrix, $\boldsymbol{H}\boldsymbol{H}^{\mathsf{T}} = d\boldsymbol{I}$, and $\boldsymbol{R}$ is a random diagonal Rademacher matrix, i.e., $\boldsymbol{R} = \mathrm{diag}(\boldsymbol{r})$, $\mathrm{P}(r_i = 1) = \mathrm{P}(r_i = -1) = 0.5$. Hence, the encoding at the i-th node is given as

$$\mathcal{C}_i(\alpha_i \boldsymbol{g}_i) = \alpha_i \boldsymbol{A}\boldsymbol{g}_i = \frac{\alpha_i}{\sqrt{m}}\boldsymbol{H}(\boldsymbol{r} \odot \boldsymbol{g}_i), \tag{5.3}$$

where fast Walsh-Hadamard algorithms can be used to perform multiplication by $\boldsymbol{H}$. Note that the edge nodes must use a common seed and follow the same random number generation protocol to generate a common random matrix for encoding.

5.4 Proposed Method: Random Linear Coding

To develop the proposed RLC, first assume that all edge nodes transmit their SG. Hence, the received signal over wireless-MAC at ER would be $\boldsymbol{y} = \sum_i \boldsymbol{A}(\alpha_i \boldsymbol{g}_i) + \boldsymbol{\eta} = \boldsymbol{A}\boldsymbol{g} + \boldsymbol{\eta}$. The node i estimates SG from received $\boldsymbol{y}$ (or its noisy version $\boldsymbol{y} + \boldsymbol{\eta}_i$) from ER via

$$\widehat{\boldsymbol{g}} = \boldsymbol{A}^{\mathsf{T}}\boldsymbol{y}. \tag{5.4}$$

Lemma 13. *$\widehat{\boldsymbol{g}}$ is an unbiased SG estimator with mean squared error*

$$\mathbb{E}\left[\|\boldsymbol{g} - \widehat{\boldsymbol{g}}\|_2^2\right] = (\frac{d}{m} - 1)\|\boldsymbol{g}\|_2^2 + d\sigma^2, \tag{5.5}$$

where σ^2 is the variance of the communication noise.[2]

We have thus far incorporated **P1** and privacy **P4** into the proposed RLC framework. Now, we take into account the constraints **P2** and **P3**, while ensuring that the estimated values at the edge nodes be an unbiased SG of $F(\cdot)$. Specifically, the developed RLC and the estimation algorithm ((5.4) or its variants) should be insensitive to the *local decisions*

[2]Note that the expectation is generally taken w.r.t. randomness in the coding, i.e., random matrix $\boldsymbol{A}$.

made at each individual node, as will be explained later.

5.4.1 Power Constraint

One major challenge in federated learning in wireless edge networks is the limited transmission power. Note that the average transmission power at node i can be computed as

$$\mathbb{E}\big[\|\boldsymbol{x}_i\|_2^2\big] = \mathbb{E}\big[\|\boldsymbol{h}_i^{-1} \odot (\alpha_i \boldsymbol{A}\boldsymbol{g}_i)\|_2^2\big] = \alpha_i^2 \|\boldsymbol{g}_i\|_2^2 \, \frac{\|\boldsymbol{h}_i^{-1}\|_2^2}{m}.$$

To control the transmission power, $\boldsymbol{x}_i$'s of all nodes can be scaled appropriately by the same value such that the transmission power constraint of all nodes are satisfied. Moreover, since the contribution of sub-channels with huge losses (small entries in $\boldsymbol{h}_i$) is remarkable in the transmission power, those sub-channels might be ignored to preserve energy at the expense of lower transmission rate. Note that the channel selection of each node in the network is performed locally and might not be known by others. Hence, it is desirable to have SG estimation at the edge nodes be independent of those local decisions. Let

$$\big[\boldsymbol{q}_i\big]_l = \begin{cases} [\boldsymbol{h}_i^{-1}]_l & \text{if sub-channel } l \text{ is being used,} \\ 0 & \text{o.w.} \end{cases} \tag{5.6}$$

To have an unbiased SG estimation given by (5.4) or its variants, we suggest scaling the transmitted signal inversely proportional to the number of channels as

$$\boldsymbol{x}_i = c\alpha_i \, \frac{m}{m_i} \big(\boldsymbol{q}_i \odot (\boldsymbol{A}\boldsymbol{g}_i)\big), \tag{5.7}$$

where m_i is the number of sub-channels being selected for data transmission by node i (i.e., $m_i = \|\boldsymbol{q}_i\|_0$ the number of non-zero entries of $\boldsymbol{q}_i$), and c is a *global* parameter shared by all nodes to control all nodes' transmission powers and may vary at different transmission

rounds. It can be easily verified that the average transmitted power at node i is

$$\mathbb{E}\big[\|\boldsymbol{x}_i\|_2^2\big] = mc^2\alpha_i^2\,\|\boldsymbol{g}_i\|_2^2\,\frac{\|\boldsymbol{q}_i\|_2^2}{\|\boldsymbol{q}_i\|_0^2}. \tag{5.8}$$

Lemma 14. *Let the transmitted signals by each edge node be given as* (5.7)*. The reconstruction given via* $\widehat{\boldsymbol{g}} = \frac{1}{c}\boldsymbol{A}^{\mathsf{T}}\boldsymbol{y}$ *provides an unbiased SG estimator. Moreover, the variance of error is bounded as*

$$\mathbb{E}\big[\|\boldsymbol{g} - \widehat{\boldsymbol{g}}\|_2^2\big] \le (\sum_i \frac{d}{m_i}\alpha_i\|\boldsymbol{g}_i\|)(\sum_i \alpha_i\|\boldsymbol{g}_i\|) - \|\boldsymbol{g}\|_2^2 + \frac{d}{c^2}\sigma^2. \tag{5.9}$$

In summary, the proposed RLC framework controls the transmitted power by appropriately adjusting c and choosing "good" sub-channels. Specifically, for a given c, to satisfy the power constraint **P2** while minimizing the MSE (5.9), it suffices to select the most number of elements from $\boldsymbol{h}_i$ with the largest magnitude such that $\mathbb{E}[\|\boldsymbol{x}_i\|_2^2]$ given via (5.8) is at most P_i. Similarly, for given m_i's (and hence $\boldsymbol{q}_i$'s), maximizing global c under the given power constraints, $\mathbb{E}[\|\boldsymbol{x}_i\|^2] \le P_i$ for all i, results in minimum MSE (5.9).

5.4.2 Transmission by a Subset of Nodes

In the wireless network, due to nodes being idle and unreliability in transmission, some nodes may not transmit their data. Let $b_i \in \{0, 1\}$ be a random variable denoting whether node i is transmitting its data at the current iteration of training or not. We assume an iid probabilistic transmission, i.e., node i transfers its data with probability π_i at each round of training, independent of other nodes, hence $b_i \sim$ Bernoulli(π_i). To compensate for this random behavior and still be able to recover an unbiased SG estimate, we propose to scale the transmitted signals by $1/\pi_i$, i.e.,

$$\boldsymbol{x}_i = c\alpha_i\,\frac{b_i}{\pi_i}\frac{m}{m_i}\big(\boldsymbol{q}_i \odot (\boldsymbol{A}\boldsymbol{g}_i)\big), \tag{5.10}$$

where $b_i = 0$ corresponds to node i not transmitting any data. Intuitively, if a node does not transmit for $\tau - 1$ round of training, at the τ-th round, its effect on the computed SG should be scaled proportionate to τ to compensate for the missing contribution in the previous rounds of training. Similar to Lemma 14, it can be shown that the reconstruction given via $\widehat{\boldsymbol{g}} = \frac{1}{c}\boldsymbol{A}^{\mathsf{T}}\boldsymbol{y}$ provides an unbiased and bounded-variance SG estimator. However, the average transmission power would be scaled by $1/\pi_i$.

Remark 6. As shown in [53], using local weighted error feedback at individual nodes can improve the convergence rate at the expense of larger memory usage at edge nodes, even for biased SG compression. Hence, by relaxing the unbiasedness constraint on RLC, for example, we can easily control the transmission power by

$$\boldsymbol{x}_i = s_i \left(\boldsymbol{q}_i \odot (\boldsymbol{A}\boldsymbol{g}_i) \right), \tag{5.11}$$

where s_i is an appropriately chosen constant, optimized locally at node i. However, to ensure convergence, the remaining portion of $\boldsymbol{g}_i$, given as $\boldsymbol{e}_i = \boldsymbol{g}_i - \frac{1}{c}\boldsymbol{A}^{\mathsf{T}}\boldsymbol{x}_i$ should be stored for transmission at later rounds of training.

5.5 Experiments and Discussions

To evaluate the performance of the proposed RLC framework, we considered training various deep models over networks of 32 and 50 nodes at different channel signal to noise ratios. Further, we assume that all nodes have the same power constraint P, and they may transmit their data with probability $\pi_i = 0.5$. For comparison, we also implemented the digital communication scheme which first compresses and encodes the stochastic gradients and then transmits the compressed values of each node one at a time. For digital data compression, we used quantized compressive sampling (QCS) [53] which provides state-of-the-art performance in terms of compression gain and convergence rate. To have the same number of channel uses (hence, the same latency per training iteration) for a network of K

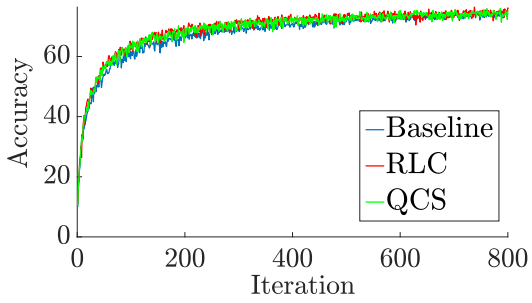
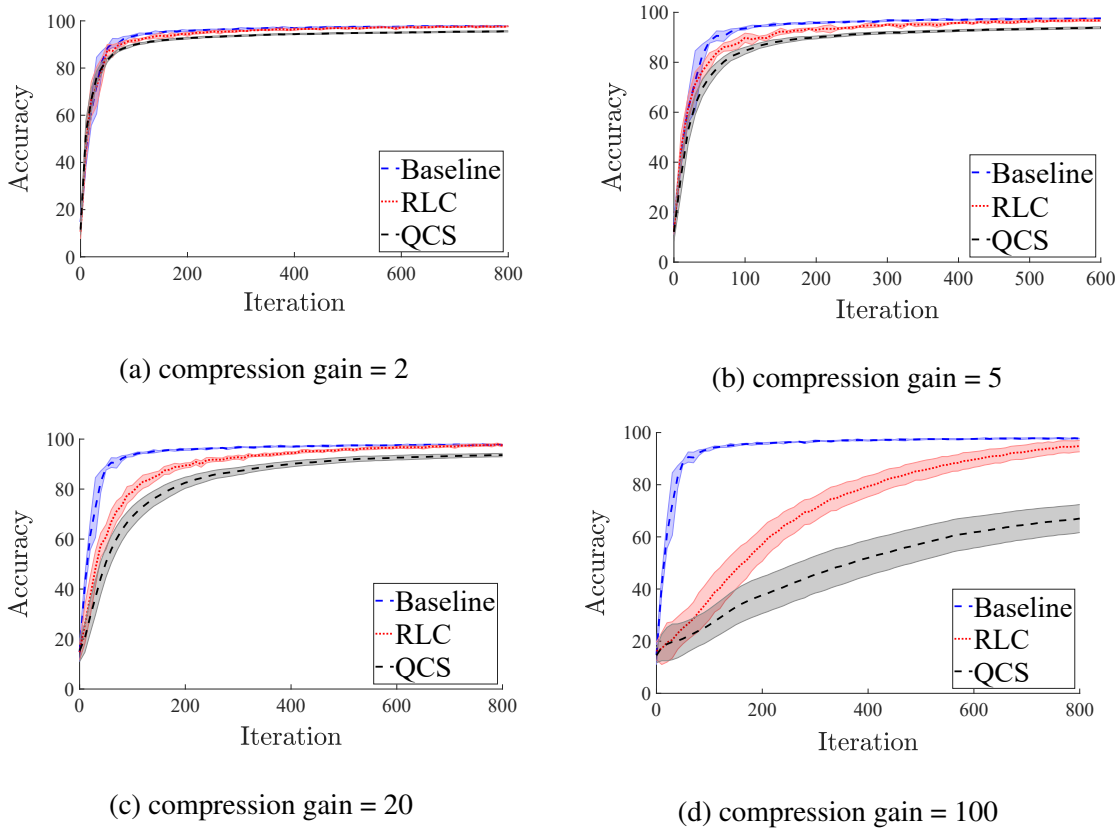


Figure 5.3: Convergence rate vs training iteration for Cifarnet. QCS has approximately 350 times more channel uses than RLC.

nodes, if the compression gain of RLC is set to be γ, the digital communication scheme have to achieve a compression gain of K_e times larger, $K_e\gamma$, where K_e is the number of non-idle transmitting nodes. Further, we optimize the parameters of QCS to achieve the minimum MSE while having the desired compression gain. We also consider baseline transmission (no SG compression and assuming infinite channel band-width). Due to the large number of nodes in the network and unbalanced distributed dataset over nodes, analog compression based on sparsity such as [110] causes large amount of distortion in the reconstructed SG, hindering the convergence of the learning algorithm.

First, we consider a network of 50 edge nodes, communicating to the ER with channel signal to noise ratio $\texttt{SNR} = 18$dB. Hence, $P/\sigma^2 \approx 63$ and the capacity of end-to-end channel is $C = 3$ bits per symbol. We then consider training Cifarnet, a deep convolutional model with approximately one million parameters, over Cifar10 dataset using stochastic gradient descent (SGD) algorithm. Traditional communication of SGs using QCS with a compression gain of 30 requires total transmission of approximately $53e6$ symbols, which results in $17.8e6$ channel uses. On the other hand, the proposed RLC framework with compression gain of 20 achieves the same performance with only $50e3$ channels uses, reducing the communication latency by a factor of at least 350. Moreover, as shown in Fig. 5.3, the convergence rate of the proposed algorithm follows that of the QCS and baseline (no SG compression) closely, *in terms of accuracy vs. number of iterations*. But since the



(a) compression gain = 2

(b) compression gain = 5

(c) compression gain = 20

(d) compression gain = 100

Figure 5.4: Convergence rate vs training iteration for Lenet over a network of 32 nodes. Baseline (blue) represents the ideal case of no SG compression and infinite communication resources.

communication latency of RLC is much lower, the *training time* using RLC is orders of magnitude smaller than digital communication.

Next, we consider training a Lenet-5 like convolutional network [94] over MNIST dataset using SGD with step-size $\mu = 0.05$. We consider different compression gains $\gamma = 2, 5, 20$ and 100 over a network of 32 nodes (with unbalanced datasets). The experiments are ran several times with different initial points to derive the mean and variance of the performance during federated learning, and are compared against QCS with the same communication requirements and Baseline (no compression and infinite communication bandwidth). As shown in 5.4, for low compression gains, the performance of training with compressed SGs are close to the baseline, although RLC slightly performs better than digital communication with QCS. However, for large compression gains, RLC outperforms QCS significantly. we

have observed similar results with different SGD step-sizes, different channel SNRs and different neural networks.

Comparing results of analog compression via RLC for federated learning over wireless-MAC with those of digital communication methods confirms that designing a compression method that utilizes the characteristics and constraints of the wireless-MAC, **P1**–**P4**, can significantly improve the convergence rate and reduce the training latency.

5.6 Conclusion

In this chapter, we considered federated learning (FL) in edge networks with communication over wireless multiple-access channels (MAC). Efficient distributed training of deep models over wireless MAC requires the communication scheme satisfy the constraints imposed by the communication medium and the network, such as unreliable transmission and idling of nodes in the network, limited transmission power, and preserving the privacy of data. Moreover, taking advantage of the characteristics of the MAC channels, such as over-the-air computations, can greatly reduce the communication latency. We developed a framework based on *Random Linear Coding* to reduce the communication overhead and training latency in FL over wireless MAC. In addition to the requirements imposed by the communication channel, we required the proposed encoding and decoding algorithms to result in an unbiased SG estimation of the deep model's cost function. This ensures that by proper adjustment of the training hyper-parameters, learning with the compressed SGs would converge. Further, we developed efficient power management and channel usage techniques to manage the trade-offs between power consumption, communication bit-rate, and convergence rate. Finally, through simulations, we showed the superior performance of the proposed method over other existing techniques.

CHAPTER 6
RESTRUCTURING, PRUNING, AND ADJUSTMENT OF DEEP MODELS FOR PARALLEL DISTRIBUTED INFERENCE

6.1 Introduction

The real-time inference and execution of many modern machine learning models has become a challenging task due to the significant increase in the complexity of deep models. Although the execution time of deep neural networks can be improved significantly by the application of parallel computing algorithms and using multiple processing units, it generally requires synchronization and data exchange among processing units to some extent. Moreover, in some real-world scenarios, such as sensor networks, the inference is done on the data observed by the entire network. However, transferring all data to a central powerful node to perform the ML task is undesirable due to the sheer amount of data to be collected, limited computational power, as well as privacy concerns. Hence, it is more favorable to develop a distributed equivalence of a trained deep model for deploying over the sensor network.

In the aforementioned applications, straightforward parallel computing algorithms cannot be arbitrarily scaled up for deep models with complex connectivity structures. Although it is possible to design deep models according to the capability and constraints of the processing system, following such an approach requires training a new deep model for every target hardware which is infeasible or demanding in many deep learning problems. As a result, in this chapter, we assume that a complex deep model has already been trained with minimum or no hardware-specific constraints on the parameters or structure of the neural network. Our goal is readjusting the model via restructuring the layers and manipulating the parameters of the neural network for more efficient parallel implementation.

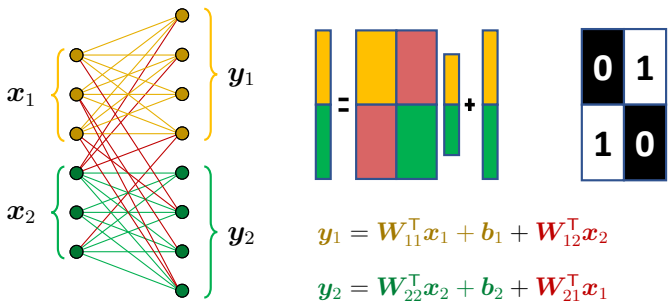


Figure 6.1: Communication between workers in parallel execution of a model over two workers. The intra-worker computations are denoted by yellow and green connections, while required communication between the workers are denoted by red edges. The binary mask matrix (right image) can be used to determine the edges between the two workers.

6.2 Problem Statement and our Approach

Consider the problem of parallel distributed implementation of a trained deep neural network over P workers, where the deep model is divided into P sub-models, each of which is executed by a worker. As managing the synchronization and data transfer among workers degrades the efficiency of the parallel implementation (e.g., higher latency), it is crucial to reduce the communication among workers. The communication is needed between the workers when the input of a neuron in a sub-model is from the output of a neuron belonging to a different sub-model which resides in another worker. These co-dependencies can lead to significant delays in computation.

For the sake of simplicity in presentations and analysis, here, we mainly focus on feedforward deep models, specifically fully-connected layers.

Consider an arbitrary neural network with L layers and parameters $\{\boldsymbol{\theta}^{(l)}\}_{l=1}^{L}$, where $\boldsymbol{\theta}^{(l)} = \{\boldsymbol{W}^{(l)}, \boldsymbol{b}^{(l)}\}$ is the parameters of the l-th layer. Let $\boldsymbol{x}^{(l)}$ be the input signal to the l-th layer. Then, the output of the layer (input to the next layer) would be given by

$$\boldsymbol{y}^{(l)} = (\boldsymbol{W}^{(l)})^\top \boldsymbol{x}^{(l)} + \boldsymbol{b}^{(l)}, \qquad \boldsymbol{x}^{(l+1)} = \sigma(\boldsymbol{y}^{(l)}), \tag{6.1}$$

where $\sigma(\cdot)$ is the activation function.

To analyze the communication bottleneck, consider an arbitrary layer with input $\boldsymbol{x}$, and parameters $\boldsymbol{W}$ and $\boldsymbol{b}$ (Fig. 6.1). Hence, $\boldsymbol{y} = \boldsymbol{W}^\top \boldsymbol{x} + \boldsymbol{b}$ would be the input signal

to the neurons of the layer. Suppose that $\boldsymbol{x}_k$ and $\boldsymbol{y}_k$ are subsets of the signals that are processed by the k-th worker. Without loss of generality, we assume that the neurons are ordered such that the k-th block of consecutive neurons belongs to the k-th sub-model, i.e., $\boldsymbol{x} = [\boldsymbol{x}_1; \boldsymbol{x}_2; \ldots; \boldsymbol{x}_P]$. By partitioning $\boldsymbol{W}$ and $\boldsymbol{b}$ accordingly, we observe that

$$\boldsymbol{y}_k = (\boldsymbol{W}_{k,k}^{\mathsf{T}} \boldsymbol{x}_k + \boldsymbol{b}_k) + (\sum_{l \neq k} \boldsymbol{W}_{k,l}^{\mathsf{T}} \boldsymbol{x}_l). \tag{6.2}$$

Note that the first term can be computed at the k-th worker independent of the others, whereas computing the second term requires synchronization and communication from the other workers. Hence, to reduce the dependency among workers and the communication cost, we consider minimizing the number of non-zero elements in $\boldsymbol{W}_{k,l}$, for $l \neq k$.

By defining an appropriate binary mask $\boldsymbol{M}$ (Fig. 6.1 (right)), the connections between sub-models can be determined by the non-zero elements of $\boldsymbol{M} \odot \boldsymbol{W}$. In general, if ι_k and o_k are the number of input and output neurons assigned to the k-th worker, then $\boldsymbol{M}$ is an anti-diagonal block matrix, given by

$$\boldsymbol{M} = 1 - \operatorname{diag}\big(\mathbf{1}_{\iota_1 \times o_1}, \ldots, \mathbf{1}_{\iota_P \times o_P}\big).$$

Remark 7. Note that the bias $\boldsymbol{b}$ does not contribute to the communication between workers and can be safely ignored in computing the cost. Further, $\|\boldsymbol{M} \odot \boldsymbol{W}\|_0$ can be viewed as the number of edges between sub-models, and be used as an approximation to the latency caused by the communication and synchronization among workers. Similarly, by defining an appropriate binary mask $\boldsymbol{M}_{ij}$, we can find the edges from worker j to i from the non-zero entries of $\boldsymbol{V}_{ij} := \boldsymbol{M}_{ij} \odot \boldsymbol{W}$. Depending on the communication protocol among workers, the number of non-zero edges, number of non-zero rows, or number of non-zero columns of $\boldsymbol{V}_{ij}$ can be interpreted as a measure of latency due to the communication from worker j to i. For the sake of simplicity, in this work, we consider $\|\boldsymbol{M} \odot \boldsymbol{W}\|_0$ as a measure of total communication latency. However, the extensions of our proposed approach to other cases is

straightforward and left as future work.

To reduce the communication, one may attempt to reduce the number of cross-edges among sub-models. However, as we observed in our experiments, generally there are many *important* connections between neurons from different sub-models, and removing those connections can severely affect the performance of the neural network. Hence, it is important to have such neurons in the same sub-model. On the other hand, the problem of neuron assignment to the workers is combinatorial and discrete with complexity $\mathcal{O}(P^N)$ for a layer with N neurons and P workers. Hence, enumerating all possibilities or using ordinary optimization techniques as well as genetic algorithms or simulated annealing would fail due to the complex nature of interactions among neurons in a deep neural network. Based on the above observations, we devise ***RePurpose***, a layer-wise neural network restructuring and pruning for efficient parallel implementation. The gist of the idea is as follows;

The neurons of the input layer are assigned to the sub-models based on each worker's computational power and/or structure of the input data. For example, in a sensor network, it is dictated by the input of each sensor. Next, we restructure and adjust the neural network, sequentially one layer at a time. For the l-th layer, the assignments of the neurons in layer $l-1$ are assumed to be fixed and known from the previous steps. The neurons in layer l are rearranged and assigned to each sub-model, and the parameters of the layer are pruned and fine-tuned, such that (i) the performance of the modified neural network is close to the original one, and (ii) the communication between the sub-models (measured by the number of edges connecting neurons from different sub-models) is minimized.

6.3 RePurpose: Restructuring and Pruning Deep Models

Consider the l-th layer of neural network and assume that the neurons in the previous layers have already been partitioned and rearranged, i.e., the input of the layer is partitioned as $[\boldsymbol{x}_1; \ldots; \boldsymbol{x}_P]$, where $\boldsymbol{x}_k$ is computed at the k-th worker. Let $\boldsymbol{y}$ and $\boldsymbol{W}$ are the signals and

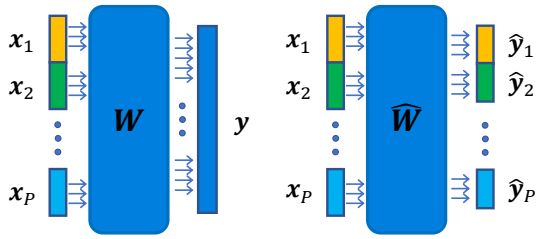


Figure 6.2: Rearranging neurons of a layer and adjusting parameters such that the k-th block of signals, $\widehat{\boldsymbol{y}}_k$, is processed at the k-th worker.

parameters of the l-th layer in the original model. RePurpose rearranges the neurons such that the k-th block of neurons are being assigned to the k-th worker (Fig. 6.2). Note that the rearrangement of the neurons can be captured via a permutation matrix $\boldsymbol{\Pi}$. Hence, if we use the same weights, the effect of neuron-rearrangement can be formulated as $\widehat{\boldsymbol{y}} = \boldsymbol{\Pi}\boldsymbol{y}$ and $\widehat{\boldsymbol{W}} = \boldsymbol{W}\boldsymbol{\Pi}^{\mathsf{T}}$, and the number of cross-edges between workers would be $\|\boldsymbol{M} \odot \widehat{\boldsymbol{W}}\|_0$. To further reduce the communication between workers, RePurpose not only rearranges the neurons, but it also prunes and adjusts $\widehat{\boldsymbol{W}}$. Hence, the optimization problem for RePurpose is formulated as

$$\min_{\widehat{\boldsymbol{W}},\boldsymbol{\Pi}} \|\boldsymbol{M} \odot \widehat{\boldsymbol{W}}\|_0 \qquad \text{s.t. } \|\widehat{\boldsymbol{W}} - \boldsymbol{W}\boldsymbol{\Pi}^{\mathsf{T}}\|_F^2 \leq \epsilon, \tag{6.3}$$

where ϵ is a parameter controlling the closeness of the parameters. Directly solving (6.3) is infeasible as it is (mixed-)discrete, non-convex, and there are $N!$ different permutation matrices. In the following, we propose an alternative and efficient approach to solve (6.3).

Recall that if neuron i is assigned to worker j, the signal at that neuron can be rewritten as $\hat{y}_i = b_i + \widehat{\boldsymbol{w}}_i^{\mathsf{T}}\boldsymbol{x} = b_i + \widehat{\boldsymbol{w}}_{ij}^{\mathsf{T}}\boldsymbol{x}_j + \sum_{k\neq j} \widehat{\boldsymbol{w}}_{ik}^{\mathsf{T}}\boldsymbol{x}_k$, where $\widehat{\boldsymbol{w}}_i$ is the i-th column of $\widehat{\boldsymbol{W}}$, and $\widehat{\boldsymbol{w}}_{ik}$ is the k-th block of $\widehat{\boldsymbol{w}}_i$ corresponding to $\boldsymbol{x}_k$. Hence, the communication cost from other workers to worker j would be $\|\widehat{\boldsymbol{w}}_{i,\setminus j}\|_0 := \sum_{k\neq j} \|\widehat{\boldsymbol{w}}_{ik}\|_0$. By incorporating an additional optional cost to encourage the total sparsity of the parameters, $\|\widehat{\boldsymbol{w}}_i\|_0$, the cost of assigning

Algorithm 3 Parameter-Space RePurpose

procedure REPURPOSEP($\boldsymbol{W}$, $\{n_k\}_{k=1}^{P}$, η_1, η_2)
    Compute the cost matrix $\boldsymbol{C}$, where $[\boldsymbol{C}]_{j,i}$ is calculated via (6.4) and (6.5).
    Construct $\widetilde{\boldsymbol{C}}$ by repeating the k-th row of $\boldsymbol{C}$, n_k times.
    $(I, J) = \text{MUNKRES}(\widetilde{\boldsymbol{C}})$
    Define permutation matrix as $\boldsymbol{\Pi}_{I,J} = 1$.
    return $\boldsymbol{\Pi}$.

neuron i to worker j would be

$$c_{ji} = \min_{\widehat{\boldsymbol{w}}_i} \|\boldsymbol{w}_i - \widehat{\boldsymbol{w}}_i\|_2^2 + \eta_1 \|\widehat{\boldsymbol{w}}_i\|_0 + \eta_2 \|\widehat{\boldsymbol{w}}_{i,\setminus j}\|_0, \tag{6.4}$$

where η_1 and η_2 control the trade-off between the error, sparsity, and cross-communication.

Lemma 15. *The solution of* (6.4) *is given by element-wise hard-thresholding* $\boldsymbol{w}_i$*, i.e.,*

$$[\widehat{\boldsymbol{w}}_i]_n = \begin{cases} 0 & |\,[\boldsymbol{w}_i]_n\,| \leq \sqrt{\eta} \\ [\boldsymbol{w}_i]_n & \text{o.w.} \end{cases} \tag{6.5}$$

where $\eta = \eta_1$ *or* $\eta_1 + \eta_2$*, depending on whether neuron* n *from the previous layer has been assigned to the* j*-th worker or not.*

Restructuring and neuron assignment can be interpreted as selecting elements from the cost matrix $\boldsymbol{C}$, whose (j, i)-th element is given by (6.4), such that (i) from row k, n_k elements are selected, i.e., n_k neurons are assigned to worker k, (ii) from each column, only one element is selected, i.e., each neuron can be assigned to only one worker, and (iii) the sum of selected elements is minimized, i.e., the total cost of neuron assignment and parameter adjustment is minimum.

Algorithm 3 summarizes the proposed solution, where MUNKRES$(\cdot)$ uses the Munkres assignment algorithm [111, 112] to find the (row-column) index pairs that minimizes the total sum cost $\sum_n [\widetilde{\boldsymbol{C}}]_{I_n, J_n}$.

Theorem 16. *Algorithm 3 finds the optimum solution of*

$$\min_{\widehat{\boldsymbol{W}},\boldsymbol{\Pi}} \|\widehat{\boldsymbol{W}} - \boldsymbol{W}\boldsymbol{\Pi}^{\mathsf{T}}\|_F^2 + \eta_1\|\widehat{\boldsymbol{W}}\|_0 + \eta_2\|\boldsymbol{M} \odot \widehat{\boldsymbol{W}}\|_0, \tag{6.6}$$

with time complexity $\mathcal{O}(N^3)$, where N is the number of layer's neurons (number of columns of $\boldsymbol{W}$).

Note that by setting $\eta_1 = 0$, (6.6) would be the Lagrangian of (6.3) and choosing appropriate value for η_2 can lead to the desired error bound $\|\widehat{\boldsymbol{W}} - \boldsymbol{W}\boldsymbol{\Pi}^{\mathsf{T}}\|_F^2 \leq \epsilon$. Finally, it is worth mentioning that the bias term does not contribute to the communication cost and is given by $\widehat{\boldsymbol{b}} = \boldsymbol{\Pi}\boldsymbol{b}$.

Remark 8. In model pruning and compression, it is common to retrain the modified model to fine-tune the parameters and improve the accuracy of the model. This extra post-processing is generally referred to as post-training phase. The same principle can be applied to our proposed algorithm.

Algorithm 4 Applying RePurpose to Deep Neural Networks

Input $\{\boldsymbol{W}^{(l)}\}_l$, $\{\boldsymbol{b}^{(l)}\}_l$, $\{n_k^{(l)}\}_{k,l}$, η_1, η_2
Output $\{\boldsymbol{\Pi}^{(l)}\}_l$, $\{\widehat{\boldsymbol{W}}^{(l)}\}_l$, $\{\widehat{\boldsymbol{b}}^{(l)}\}_l$
$\boldsymbol{E} = \eta_1 + \eta_2\boldsymbol{M}$
$\boldsymbol{\Pi}^{(0)} \leftarrow \boldsymbol{I}$
for layers $l = 1, \ldots, L$ **do**
    $\boldsymbol{T} \leftarrow \boldsymbol{\Pi}^{(l-1)}\boldsymbol{W}^{(l)}$
    $\boldsymbol{\Pi}^{(l)} \leftarrow \text{RePurpose}(\boldsymbol{T}, \{n_k^{(l)}\}_k, \eta_1, \eta_2)$
    $\widehat{\boldsymbol{W}}^{(l)} \leftarrow \mathcal{H}_{\boldsymbol{E}}\big(\boldsymbol{T}(\boldsymbol{\Pi}^{(l)})^{\mathsf{T}}\big)$
    $\widehat{\boldsymbol{b}}^{(l)} \leftarrow \boldsymbol{\Pi}^{(l)}\boldsymbol{b}^{(l)}$

Remark 9. Recall that when applying RePurpose to layers of a neural network, permuting neurons of layer l with matrix $\boldsymbol{\Pi}$ changes the signal of that layer by $\widehat{\boldsymbol{y}}^{(l)} = \boldsymbol{\Pi}\boldsymbol{y}^{(l)}$ and affects the weight matrix of that layer by $\boldsymbol{W}^{(l)}\boldsymbol{\Pi}^{\mathsf{T}}$. As a result, $\widehat{\boldsymbol{x}}^{(l+1)} = \boldsymbol{\Pi}\boldsymbol{x}^{(l+1)}$ and to have the same signal at the next layer, $\boldsymbol{y}^{(l+1)}$, the weight matrix of layer $l+1$ should be modified as $\boldsymbol{\Pi}\boldsymbol{W}^{(l+1)}$. The detailed application of RePurpose to a deep neural network with weights and

biases $\{\boldsymbol{W}^{(l)}, \boldsymbol{b}^{(l)}\}$ is presented in Algorithm 4, where $\mathcal{H}_{\boldsymbol{E}}(\cdot)$ is the (modified) element-wise hard-thresholding operator, defined as

$$\boldsymbol{Y} = \mathcal{H}_{\boldsymbol{E}}(\boldsymbol{X}): \quad Y_{ij} = \begin{cases} 0 & \text{if } |X_{ij}|^2 \leq E_{ij} \\ X_{ij} & \text{o.w.} \end{cases} \tag{6.7}$$

6.4 Performance of RePurposed Model

To analyze the performance of the modified neural network, assume that the original neural network has the following properties:

A1. The activation functions are 1-Lipschitz, i.e., for all u, v, $|\sigma(u) - \sigma(v)| \leq |u - v|$.

A2. The Frobenius norms of the weights of the neural network are bounded, i.e., for some constant $\tau > 0$, $\|\boldsymbol{W}^{(l)}\|_F \leq \tau$, for all layers $l = 1, \ldots, L$.

A3. The signals in the neural networks are bounded, i.e., there exists a constant $B > 0$ such that for input signal $\boldsymbol{x}^{(1)} = \boldsymbol{x}_{in}$, and forward signals $\{\boldsymbol{x}^{(l)}\}_{l=2}^{L}$ (outputs of the hidden layers), $\|\boldsymbol{x}^{(l)}\|_2 \leq B$ for $l = 1, \ldots, L$.

Moreover, suppose that the parameters η_1 and η_2 at each call of the REPURPOSE are adjusted such that the solution of Lagrangian formulation (6.6), given by REPURPOSE, is also the solution of the following constrained optimization problem

$$\min_{\widehat{\boldsymbol{W}}, \boldsymbol{\Pi}} \|\boldsymbol{M} \odot \widehat{\boldsymbol{W}}\|_0 \qquad \text{s. t. } \|\widehat{\boldsymbol{W}} - \boldsymbol{T}\boldsymbol{\Pi}^{\mathsf{T}}\|_F^2 \leq \epsilon. \tag{6.8}$$

Hence, by Alg. 4 and the cascade application of RePurpose, the modified weight matrix of the l-th layer of neural network satisfies $\|\widehat{\boldsymbol{W}}^{(l)} - \boldsymbol{\Pi}^{(l-1)}\boldsymbol{W}^{(l)}(\boldsymbol{\Pi}^{(l)})^{\mathsf{T}}\|_F^2 \leq \epsilon$. For the simplicity in notations, let $\varepsilon = \sqrt{\epsilon}$.

Theorem 17. *For an input data $\boldsymbol{x}$, let $\boldsymbol{y}$ and $\widehat{\boldsymbol{y}}$ be the outputs of the original and RePurposed neural network, respectively. If $\boldsymbol{\Pi}$ is the permutation of the final output neurons in the*

*RePurposed neural network, then under assumptions **A1-3**,*

$$\|\widehat{\boldsymbol{y}} - \boldsymbol{\Pi y}\|_2 \leq \varepsilon \frac{(\tau+\varepsilon)^L - 1}{\tau + \varepsilon - 1} B. \tag{6.9}$$

Especially, if the parameters of the neural network are normalized such that $\|\boldsymbol{W}^{(l)}\|_F = 1$, then $\|\widehat{\boldsymbol{y}} - \boldsymbol{\Pi y}\|_2 \leq \big((1+\varepsilon)^L - 1\big)B$.

Therefore, if the hyperparameters of RePurpose are chosen carefully, we can ensure that the output of the modified neural network is close to the original model (after accounting for the possible rearrangement of the neurons of the output layer).

6.5 Experiments

To evaluate the performance of the RePurpose algorithm, we consider different neural network architectures and compare the accuracy, communication and wall-clock times w.r.t. *naive* implementation where the input data is communicated to all nodes in the network, so they all have the entire input data, *baseline* which is parallel implementation of the deep model without any modification to the parameters or structures, and *sparse* implementation which sparsifies the parameters to reduce cross-edges between the workers without re-arranging the neurons. We evaluate the accuracy-communication trade-off in different sensor networks, as well as the reduction in total computation time (wall-clock time) in Edge and Data Center platforms.

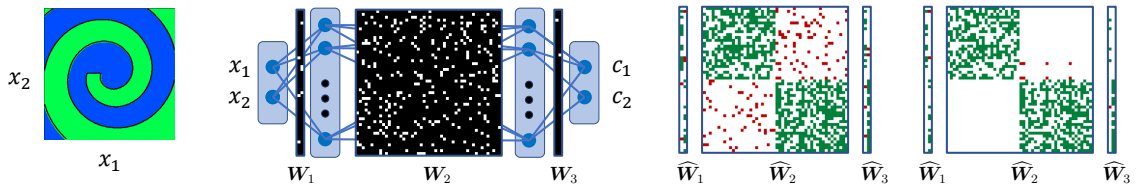
6.5.1 Sensor Network

Setup 1. Figure 6.3(a) shows a 2 sensors network, sensor i observes location x_i of a target object and each sensor's task is to determine whether the object is in the blue or green region. A simple neural network (Fig. 6.3(b)) is trained at a central node to perform the task with accuracy 94.5%. In the naive approach, the sensors exchange their observations (x_i's) and run the inference (NN) independently. Hence, the NN is executed twice throughout

the network at the cost of higher computational complexity. Alternatively, we can apply RePurpose to efficiently distribute the inference over the sensors. We applied RePurpose with $\eta_1 = 0$, $\eta_2 = 0.01$ (Fig. 6.3(c)), and $\eta_2 = 0.1$ (Fig. 6.3(d)). As a result, the number of cross-worker communications reduced significantly to 1.7%, 1.5% and 1.6% for $\eta_2 = 0.01$, and 0.7%, 0.1% and 0.3% for $\eta_2 = 0.1$ for layers 1, 2, and 3, respectively. Specifically, with only 6 communicated values, the computational complexity at each sensor is reduced by almost a factor of 4 compared to the naive implementation. However, the accuracy of the distributed parallel model, prior to the post-training phase, is reduced to 93.5%. By retraining the modified model for few iterations (and imposing the structural constraints found through RePurpose), the accuracy of the fine-tuned model becomes 94.4%.

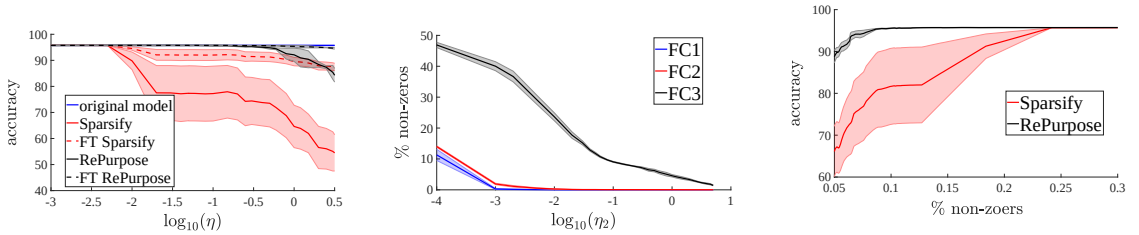
Setup 2. Next, we consider a network of P sensors where each sensor observes an image of a digit x_i (from MNIST dataset) and the goal is finding the rounded average $\left[\frac{\sum_i x_i}{P}\right]$. We adapted a Lenet-5 like structure [94] for the neural network which is trained in a central server (Fig. 6.6), and repeated the experiments several times. Note that one might attempt to classify the digits at each individual sensor and then share the value with other nodes to compute the average. However, in addition to the increased computational complexity at each individual node, it is worth mentioning that if the accuracy of digit recognition is ρ, close to 1, then the final accuracy in computing the average would be approximately $\frac{1+8\rho^P}{9}$. For example, for a network with 6 nodes and $\rho = 0.98$, the final accuracy would be less than 90%. We applied the RePurpose algorithm on the trained model for distributed inference over the sensor network with different communication (cross-worker edges) constraints. Fig. 6.4 compares the results of RePurpose with the baseline and direct sparsification, in a network with $P = 6$ sensors.

Setup 3. Next, we consider P sensors (cameras) that observe a scene and detect whether an specific object exists or not. For this purpose, we used a Resnet-like neural network [113] over CIFAR10 and the objective is detecting the presence of a "dog" in any of the images (Fig. 6.7). Fig. 6.5 shows the results of RePurpose, the baseline, and direct sparsification, in



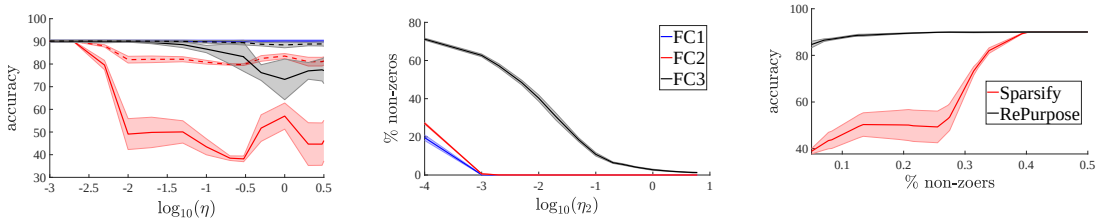
(a) Classification Regions (b) Original trained model (c) RePurpose $\eta_2 = 0.01$ (d) RePurpose $\eta_2 = 0.1$

Figure 6.3: **Setup 1.** Distributed inference over a sensor network to classify location of an object. The zero coefficients are represented by empty (white) spaces, inner-worker connection by green pixels and cross-worker edges by red pixels in the images. Note that for the illustration purposes, the coefficient matrix of the first layer is transposed.



(a) Accuracy vs η_2 (b) % of cross-edges vs η_2 (c) Accuracy vs % of cross-edges

Figure 6.4: RePurpose vs Sparsification, a network with 6 workers in **Setup 2**



(a) Accuracy vs η_2 (b) % of cross-edges vs η_2 (c) Accuracy vs % of cross-edges

Figure 6.5: RePurpose vs Sparsification, a network with 2 workers in **Setup 3**

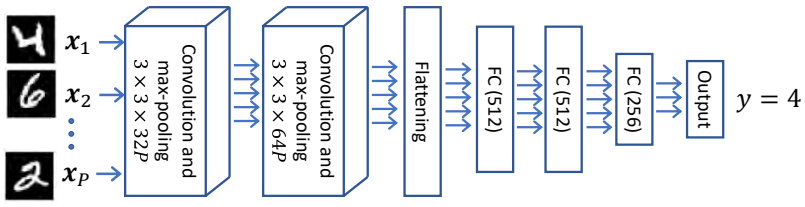


Figure 6.6: Structure of CNN for **Setup 2**

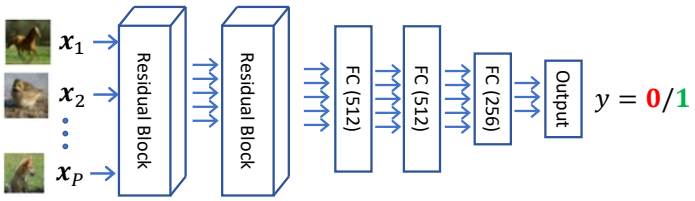


Figure 6.7: Structure of neural network for **Setup 3**

a network with $P = 2$ sensors.

In figures 6.4a and 6.5a, we verified the effect of the parameter η_2 in (6.6) on the accuracy of the restructured and pruned model. By increasing η_2, i.e., decreasing the number of cross-communication among workers, the accuracy of the distributed model decreases. As seen from the figures, RePurpose significantly outperforms direct sparsification in terms of th accuracy of the modified model. Although the accuracy of the modified model is dropped for large η_2 (i.e., extremely low cross-communications), with 1 or 10 epochs of post-training for MNIST and CIFAR10, respectively, ("FT RePurpose" in the figures) it achieves almost the same accuracy as the original model, while direct sparsification fails to provide good accuracy. Figures 6.4b and 6.5b shows the number of cross-communication among workers versus η_2 for the individual hidden layers of the considered neural networks. Interestingly, RePurpose sparsifies the cross-edges between workers significantly for the hidden layers. The restructured model can achieve the same performance as the original model by using less than 0.0003 of the cross-edges (i.e., between 10 to 30 connections out of more than 100000 edges between workers). Finally, figures 6.4c and 6.5c compare the accuracy vs the cross-communication between workers. Clearly, direct sparsification performs well *only* when there is sufficient cross-communication among the workers, while the accuracy of the model obtained by RePurpose does not change for a vast sparsity range.

Finally, it is worth mentioning that in the naive approach to inference over the sensor network, each node has to transmit its observations to other nodes, hence the communication between any two pair of nodes would be 784 or 1024 values for **Setups 2** and **3**, respectively. However, RePurpose can achieve the same accuracy with *exchanging less than total of 200*

Table 6.1: Target Accelerator Evaluation Platforms

Name	Node Compute	Node Memory	Network Bandwidth	Number of Nodes
Datacenter	125 TOPS	4GB	150 GB/s (NVLink)	1-32
Edge	0.5 TOPS	1GB	100 MB/s (Ethernet)	1-32

values across the entire network.

6.5.2 System Evaluations

Methodology- We evaluate RePurpose on two distributed accelerator platforms, described in Tbl. 6.1, simulated using ASTRA-sim [114]. ASTRA-sim is an open-source distributed Deep Learning platform simulator that models cycle-level communication behavior in details for any partitioning strategy across multiple interconnected accelerator nodes. ASTRA-sim takes the compute cycles for each layer of the model as an external input, and manages communication scheduling similar to communication libraries like NVIDIA NCCL [115]. We obtained compute cycles for the Datacenter configuration from a NVIDIA V100 GPU implementation, and for the Edge configuration (e.g., sensor network) from a separate DNN accelerator simulator [116].

We tried to stress the aforementioned platforms under various sized problems to show the efficiency of RePurpose. In all models, we assumed a stack of 5 layers with same number of neurons. In our notation, N refers to the number of neurons per layer (or matrix dimensions). For the datacenter system, N varies from $1K$ to $1M$, while for edge system the variation is from $1K$ to $32K$. We also assumed strict ordering between current communication and computation of next layer, meaning that each node begins computation of each layer only when it has all inputs available.

We picked 4 different flavors of RePurpose with 50%, 75%, 90% and 99% sparsity factor named as RP-50, RP-75, RP-90, and RP-99, respectively. In addition, we changed the number of worker nodes from 1 to 32 for both system configurations.

Results- Fig. 6.8 shows the total amount of data that each node needs to send out for

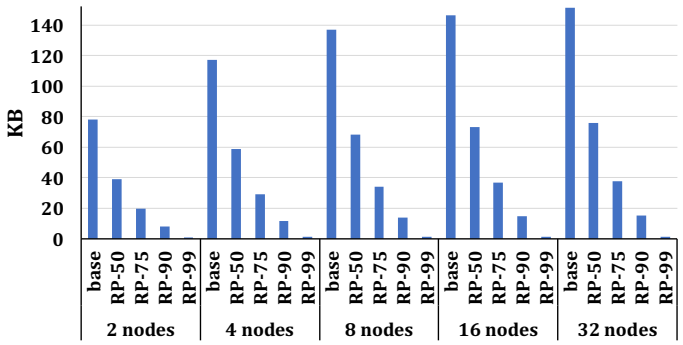
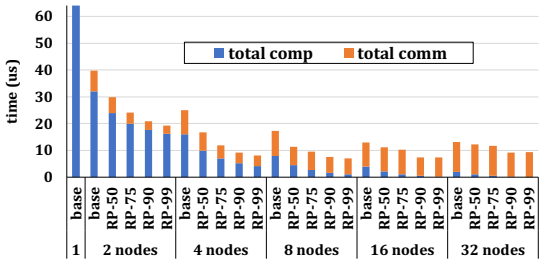


Figure 6.8: Theoretical amount of data each node needs to send out for $N = 8K$.

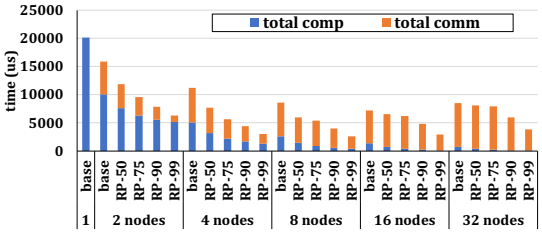
one input sample for $N = 8K$. Clearly, specification has the linear effect on the amount of communicating data. On the other hand, partitioning across more nodes also increases the total communicating data. But the increase in rate diminishes as the number of nodes increases, converges to 2X more data compared to the case of 2 nodes.

To further investigate the effect of RePurpose in reducing the computation and communication times, Fig. 6.9 shows the simulation results of the communication and computation breakdown for the baseline system and RePurpose for $N = 8k$. As seen from Fig. 6.9a, in a datacenter system, on average and across different number of nodes, RP-50, RP-75, RP-90 and RP-99 achieve 1.7×, 2.76×, 4.77× and 10.47× speed-up in computations, respectively. The average improvement for communication ratio is 1.2×, 1.45×, 1.74× and 1.75×, respectively. The reason for lower improvements of communication time is that due to NVLink's high bandwidth. For $N = 8K$, network communication time is mostly network latency limited. Hence, reduction in input size does not correspond to linear reduction in communication time.

Fig. 6.9b shows the similar results but for edge system. Here, due to much lower network bandwidth, the effect of communication is more considerable. On average applying RP-50, RP-75, RP-90 and RP-99 improve computation times by 1.7×, 2.77×, 4.78× and 11.01×, respectively. This value for communication is 1.2×, 1.38×, 1.82× and 3.04× respectively. As the number of nodes grow, the communication gap between the baseline and RePurpose decreases. This is mostly because of the congestion in the network (e.g. switch) that

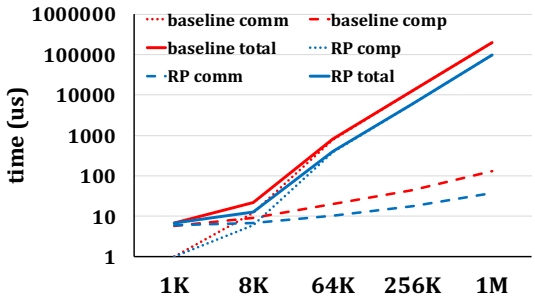


(a) Datacenter Platform results

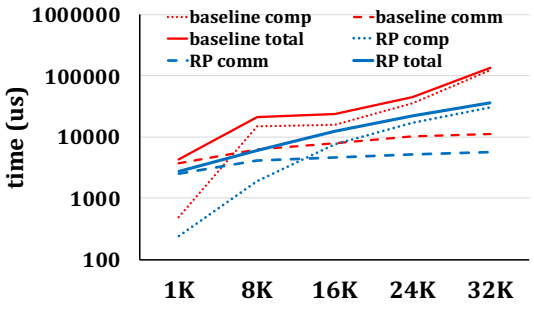


(b) Edge Platform results

Figure 6.9: Communication and computation breakdown across different systems and $N = 8K$



(a) Datacenter Platform results



(b) Edge Platform results

Figure 6.10: The effect of communication vs. computation times as the model size N grows

decreases the effect of benefits gained by RePurpose.

Fig. 6.10 shows how communication, computation and total times change as the the number of neurons grows. For each network size, computation and communication times are averaged across different sparsity factors and node counts. For datacenter system (Fig6.10a), computation is the dominant factor. This is expected since the computation grows as $O(N^2)$ while communication increases as $O(N)$. Since the network band-width is very high in datacenter, the effect of communication is negligible. In general, the total time ratio increase from $1.01\times$ in $N = 1K$ to $2.06\times$ in $N = 1M$. On the other hand, communication remains a considerable factor in the edge systems (Fig. 6.10b) due to: (i) low network bandwidth, and (ii) lower dimensions of workloads on edge systems. The total time improvement for edge system is $1.55\times$ for $N = 1K$ and it increases to $3.8\times$ for $N = 32K$.

6.6 Conclusion

In this chapter, we considered the problem of efficient parallel distributed inference of an already trained deep model over a cluster of processing units or a sensor network. Required communication and synchronization among processing units or network nodes (i.e., workers) can adversely affect the computation time. Moreover, in the wireless sensor networks, it may significantly increase the power consumption due to the transmission of large amounts of data. We claimed that traditional approaches to prune or compress the deep models fail to consider the constraints imposed in such distributed inference systems. To overcome the shortcomings of the existing methods, we devised *RePurpose*, a framework to restructure the deep model by rearranging the neurons, optimum assignment of neurons to the workers, and then pruning the parameters, such that the dependency among workers is reduced. To efficiently solve RePurpose, we used ℓ_0 optimization and the Munkres assignment algorithm. We showed that RePurpose can significantly reduce the number of cross-communication between workers and improve the computation time significantly, while the performance loss of the modified model is negligible.

CHAPTER 7
CONCLUSION

7.1 Summary of Achievements

In recent year, the complexity of deep learning problems has increased significantly, both in terms of the number of parameters and the available training data samples. Hence, training or real-time inference of modern deep models on end-user devices or a single processing node is unappealing or nearly impossible due to the required storage, memory or computational power. In this dissertation, we investigated challenges in distributed training and inference of deep models. As communication overhead is a major bottleneck in such distributed systems, in my research, we focused on reducing the required communication among workers to improve the convergence rate in distributed deep learning or execution time during inference.

In the first part of my dissertation, we considered distributed deep learning. To reduce the communication overhead, we developed and analyzed various algorithms from three different perspectives: Information Theory and Central Estimation Officer (CEO) problem, matrix factorization, and compressive sampling.

In chapter 2, we framed distributed learning as a CEO problem. We argued that the computations at the workers can be considered as noisy observations of the true update (or gradient), $\boldsymbol{\theta}^*$, and the objective of distributed learning would be reliable estimation of $\boldsymbol{\theta}^*$ with minimum communication from workers. Based on this principle, we developed and analyzed a framework for distributed learning. The proposed method consists of three major blocks: 1) dithered and nested quantization at the workers, 2) distributed source coding to incorporate the correlation among workers for further reduction in communication bit rate, and 3) decoding the data received from the workers and estimating the optimum parameters at the server. Via simulations, we showed that the proposed method reduces the

communication bit-rate compared to the other existing methods while it can achieve nearly the same convergence speed as of the baseline training [51, 50].

In chapter 3, we considered compressing the stochastic gradients via low-rank matrix factorization and then quantizing the factorized terms. However, naïvely pursuing such an approach in distributed machine learning is costly, both in terms of the computations and the resulting error during training. We advocated using the factorization inherently provided during the backpropagation algorithm. However, traditional quantization of these factors results in biased compression and large MSE, which adversely affect the convergence of distributed learning. We devised two novel Indirect SG Quantization (ISGQ) methods with the ultimate goal of providing an unbiased SG compression with minimum error in the reconstructed SG. We showed that with the same number of quantization levels, the MSE of ISGQ is comparable to or better than Lloyd-Max direct quantization of SGs which translates into better convergence rates. Moreover, ISGQ has less computational complexity than traditional quantization schemes, and it can achieve compression gains of more than 100, while the compression gain of the existing quantization methods is at most 32 [52].

In chapter 4, we considered the problem of achieving an arbitrarily large *unbiased* compression of SG while ensuring the mean squared error (MSE) is low. Inspired by the design of structured mixing matrices in compressed sensing, we developed Quantized Compressive Sampling (QCS) and showed that it can achieve orders of magnitude smaller MSE compared to the state-of-the-art unbiased compression techniques, resulting in superior convergence rate. More precisely, for non-convex optimization problems, stochastic gradient descent with the proposed QCS compression enjoys the same convergence rate of $\mathcal{O}(1/\sqrt{T})$ as the baseline training, where T is the number of training iterations. However, since the constant factor in the rate is slightly larger due to the compression, there is an $\mathcal{O}(1/\sqrt{T})$ gap w.r.t. the baseline training. We showed that utilizing *weighted* error feedback reduces the effect of SG compression on convergence rate to $\mathcal{O}(1/T)$ [53].

Next, in chapter 5, we considered federated learning in wireless edge networks. Efficient

communication requires the compression algorithm to satisfy the constraints imposed by the communication medium and take advantage of its characteristics, such as over-the-air computations inherent in wireless multiple-access channels, unreliable transmission and idle nodes in the edge network, limited transmission power, and preserving the privacy of data. To achieve these goals, we proposed a novel framework based on Random Linear Coding and developed efficient power management and channel usage techniques to manage the trade-offs between power consumption, communication bit-rate, and convergence rate of federated learning over wireless MAC. We showed that the proposed encoding/decoding results in an unbiased compression of SG, hence guaranteeing the convergence of the training algorithm without requiring error-feedback. Finally, through simulations, we showed the superior performance of the proposed method over other existing techniques [62].

Finally, in chapter 6, we investigated the problem of distributed parallel inference and how to modify the structure or parameters of already-trained deep neural networks to make them suitable for efficient deployment on target platforms. To reduce the latency due to the communication across workers in distributed inference, we proposed to rearrange the neurons in the neural network and partition them (without changing the general topology of the neural network), such that the interdependency among sub-models is minimized under the computations and communications constraints of the workers. We developed RePurpose, a layer-wise model restructuring and pruning technique that guarantees the performance of the overall parallelized model. To efficiently solve RePurpose, we used ℓ_0 optimization and the Munkres assignment algorithm. We showed that, compared to the existing methods, RePurpose significantly improves the efficiency of the distributed inference via parallel implementation, both in terms of communication and computational complexity [89].

7.2 Future Research Directions

7.2.1 Distributed Learning and Inference over Graphs

In this dissertation, we have assumed that all workers can communicate with each other or a central node. However, in many networks, the communication among workers has to be done in a certain order or follow the topology of the network. An immediate research problem would be how to adjust or extend our results to arbitrary decentralized graph networks. For example, since each node can communicate only with its neighbors, it is more desirable to have model update (in distributed learning) or computations (in distributed inference) at a node depend solely on the neighbors. However, extension of the proposed algorithms such as quantized compressive sampling to graph networks remained as an open research problem. Moreover, in RePurpose, we implicitly assumed that all nodes can communicate with each other and considered the total number of communications in the network as a measure of latency. Extending RePurpose to more general networks and analyzing its performance would be a topic of future research.

7.2.2 Robust Distributed Inference

In this thesis, we assumed that all nodes are robust against failure and they synchronously communicate with each other or a central node whenever it is required. However, in practice, some nodes may have high computational latency or susceptible to failures. Hence, results from a subset of nodes may not be available in time. In recent years, inspired by the results in channel coding for error correction, it was proposed to incorporate some redundancy into the computations, such that by receiving results from only a random fraction of workers, the desired objective can be computed. As an example, computing $\boldsymbol{y} = [\boldsymbol{W}_1; \boldsymbol{W}_2]\boldsymbol{x}$ can be distributed as $\boldsymbol{y}_1 = \boldsymbol{W}_1\boldsymbol{x}$, $\boldsymbol{y}_2 = \boldsymbol{W}_2\boldsymbol{x}$ and $\boldsymbol{y}_3 = (\boldsymbol{W}_1 + \boldsymbol{W}_2)\boldsymbol{x}$ over 3 workers. By receiving computations from any two workers, the server can easily compute $\boldsymbol{y}$. As a possible future research direction, we suggest incorporating redundancies when restructuring complex deep

models and distributing the computations across a sensor/processing network, such that not only it satisfies the communication and computational constrains of the network, but also is robust against node failures.

7.2.3 Distributed Model Training

Sometimes the deep model is too large to fit in a single node's RAM memory to execute the training algorithm, or the deep-model training could be computationally too heavy to be done at a single node in a reasonable amount of time. To overcome the challenges, we propose a new paradigm; distribute the complex deep model across different workers such that the number of parameters and the complexity of each sub-model would not exceed the computational capabilities of the workers. A fundamental question that arises is how we may train such a network as each worker possesses only a small subset of the model but its parameters' updates depend on the model parameters of other workers.

Ideally, it is desired to update each worker's sub-model without any communication from other workers. As an example, consider the linear regression problem, $\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x}$. If worker k has only a subset Ω_k of coefficients $\boldsymbol{W}$, i.e., $\boldsymbol{W}\big|_{\Omega_k}$, and set the rest to zero, then we have shown that knowing some statistical properties of input $\boldsymbol{x}$ can help to find the optimum $\boldsymbol{W}_o$ for linear regression from partially computed $\boldsymbol{W}\big|_{\Omega_k}$'s after the local sub-models are converged. Hence, distributed model training can be achieved with only one round of communications. However, for more complex deep models, it remains open as how to merge the locally computed partial parameters and what kind of extra information is required.

Appendices

APPENDIX A
DITHERED QUANTIZATION

It is well-known that the error in ordinary quantization, especially when the number of quantization levels is low, depends on the input signal and is not necessarily uniformly distributed. In *Dithered Quantization*, a (pseudo-)random signal called dither is added to the input signal prior to quantization. Adding this controlled perturbation can cause the statistical behavior of the quantization error be more desirable [91, 102, 117].

Let $Q(\cdot)$ be an M-level uniform quantizer with quantization step size of Δ, i.e., $Q(v) = \Delta \lfloor v/\Delta \rceil$ and the output range of $Q(\cdot)$ is $\{-M, \ldots, 0, \ldots, M\}$.[1] The dithered quantizer is defined as follows;

Definition (Dithered Quantization)**.** For an input signal x, let u be a dither signal, independent of x. The dithered quantization of x is defined as $\tilde{x} = Q(x + u) - u$.

Remark 10. To transmit the dithered quantization of x, it is sufficient to send the index of the quantization bin that $x + u$ resides in, i.e., $\lfloor (x + u)/\Delta \rceil$. The receiver reproduces the (pseudo-)random sequence u using the same random number generator algorithm and seed number as the sender. It is then subtracted from $Q(x + u)$ to reconstruct $\tilde{x}$.

Theorem 18 ([91])**.** *If 1) the quantizer does not overload, i.e., $|x + u| \leq \frac{M\Delta}{2}$ for all input signals x and dither u, and 2) The characteristic function of the dither signal, defined as $M_u(j\nu) = \mathbb{E}_u[e^{j\nu u}]$, satisfies $M_u(j\frac{2\pi l}{\Delta}) = 0$ for all $l \neq 0$, then the quantization error $e = x - \tilde{x}$ is uniform over $(-\Delta/2, \Delta/2]$ and it is independent of the signal x.*

It is common to consider $\mathcal{U}(-\Delta/2, \Delta/2)$ as the distribution of the random dither signal which can be easily verified that it satisfies the conditions of Thm. 18.

[1] Throughout the paper, we assume that all quantizers are centered around 0 (with the exception of sign-based quantization). This is the case also for ternary [16] and stochastic quantization [15].

In some cases, the receiver may not be able to reproduce the dither signal to subtract from $Q(x+u)$. Hence, quantization is simply defined as $\tilde{x}_h = Q(x+u)$. We refer to this approach as the *half-dithered quantization* as the dither signal is applied only to the quantization, not the reconstruction of x. In this case, the quantization error is not necessarily independent of the signal, however by an appropriate choice of the dither signal, the moments of the quantization error will be independent [102]. For example, if the dither signal u is the sum of k independent random variables, each having uniform distribution $\mathcal{U}(-\Delta/2, \Delta/2)$, then the k-th moment of the quantization error, $\epsilon = x - \tilde{x}_h$, would be independent of the signal; $\mathbb{E}\left[\epsilon^k | x\right] = \mathbb{E}\left[\epsilon^k\right]$.

Remark 11 (1-Bit Dithered Quantization). Note that the output range of the dithered quantization is $\{-M, \ldots, +M\}$. Hence, each value is represented by minimum of $\log_2(1+2M)$ bits (without applying any compression to the quantized sequence). Reducing the number of bits to only 1-bit while keeping the desired properties of the dithered quantizers can potentially reduce the transmission bits by almost 50% (from at least $\log_2 3 \approx 1.58$ bits to 1 bit).

Without loss of generality, assume that $|x| \leq 1/2$. We propose the following dithered 1-bit quantization:

$$q = \text{sign}(x+u) := \begin{cases} +1 & \text{if } u + x > 0 \\ -1 & \text{o.w.} \end{cases}, \tag{A.1}$$

where $u \sim \mathcal{U}(-1/2, 1/2)$ is the random dither signal. The dequatized value is then given by

$$\tilde{x} = q - u. \tag{A.2}$$

It is straightforward to show that this 1-bit dithered quantizer is unbiased and the quantization

noise is uniformly distributed and independent of x;

$$\mathbb{E}[\tilde{x} - x] = 0, \quad \mathrm{Var}[\tilde{x} - x] = \frac{1}{12}. \tag{A.3}$$

Relationship with Ternary and Stochastic Quantizations

Without loss of generality, assume that the vector $\boldsymbol{x}$ is normalized such that $|x_i| \leq 1$. Although the reconstruction of quantized values in our method is different from those in TernGrad and QSGD, we show that these quantizers can be considered as a special case of the half-dithered quantizer.

M-level Stochastic Quantization in [15] is defined as

$$Q^{(s)}(x_i) = \begin{cases} \mathrm{sign}(x_i)\,\frac{l}{M} & \text{with prob. } 1 - d(x_i) \\ \mathrm{sign}(x_i)\,\frac{l+1}{M} & \text{with prob. } d(x_i) \end{cases} \tag{A.4}$$

where l is the quantization bin that $|x_i|$ resides in, i.e., $|x_i| \in [l/M, (l+1)/M]$ and $d(x_i) := M|x_i| - l$. The ternary quantizer of [16] can be considered as a special case of stochastic quantizer with $M = 1$.

Lemma 19. *Stochastic quantization is the same as* $(2M+1)$*-level half-dithered quantizer with step-size* $\Delta = \frac{1}{M}$ *and uniform dither* $u \sim \mathcal{U}(-\frac{1}{2M}, \frac{1}{2M})$.

In other words, stochastic quantizer adds a uniformly distributed dither to the input signal before quantization. However, the receiver *does not* subtract the dither from the quantized value. Therefore, the quantization error is not independent of the signal. It can be easily verified that although the quantization is unbiased, $\mathbb{E}\big[\boldsymbol{x} - Q^{(s)}(\boldsymbol{x})\big] = \mathbf{0}$, its variance depends on the value of the input signal and varies in $[0, 1/4M^2]$, depending on the input signal x;

$$\mathbb{E}\big[([Q^{(s)}(\boldsymbol{x}) - \boldsymbol{x}]_i)^2\big] = \frac{d(x_i)(1 - d(x_i))}{M^2}.$$

On the other hand, the variance of the dithered quantization noise would be uniformly

$1/12M^2$, independent of x. For example, if x is uniformly distributed over $[-1, 1]$, the average quantization variance of the stochastic quantizer would be $1/6M^2$, twice the variance of the dithered quantization.

APPENDIX B

STATISTICAL PROPERTIES OF THE SIGNALS IN ISGQ

Here, we focus on the neural networks with ReLU activation functions as it is the most commonly used one in modern deep models. Consider the l-th layer of a neural net. For a layer with large enough input nodes (dimension of input signal)[1], we can approximate the distribution of the input signals to the nodes, $\boldsymbol{y}^{(l)} = \boldsymbol{W}^{(l)}\boldsymbol{x}^{(l-1)} + \boldsymbol{b}^{(l)}$, as multivariate Gaussian. It is worth mentioning that in practice, although the hidden layers' outputs are generally sparse, but the random-like behavior of signals and weights [118] justifies the Gaussian behavior as verified by our simulations. Hence, the distribution of the output, $\boldsymbol{x}^{(l)} = \max(\boldsymbol{y}^{(l)}, \boldsymbol{0})$, will have a peak at 0 with an approximately Folded Normal distribution for positive values.

Similarly for the backward signals, we note that if $y_j^{(l)} > 0$, $\delta_j^{(l)} = \sum_k w_{k,j}^{(l+1)}\delta_k^{(l+1)}$ would be the (weighted) sum of multiple signals from the next layer. Thus, we may approximate its distribution as Normal. On the other hand, for $y_j^{(l)} < 0$, $\delta_j^{(l)} = 0$. As the neural network trains and converges to a (local) minimum solution, the gradients and therefore, the backward signals become mostly zero or insignificant.

Remark 12. For the Softmax layer with cross-entropy cost function, commonly used in classification tasks, $\boldsymbol{\delta}^{(out)} = \boldsymbol{x}^{(out)} - \boldsymbol{d}$, where $\boldsymbol{d} \in \{0, 1\}$ is the desired output. Since $0 \leq \boldsymbol{x}^{(out)} \leq 1$, $\boldsymbol{\delta}^{(out)} \in [-1, 1]$. At the initial stages of training, the classification algorithm behaves randomly and we assume that $\boldsymbol{\delta}^{(out)} \sim \mathcal{U}[-1, 1]$. However, as the network is trained and its accuracy improves, $\boldsymbol{x}^{(out)}$ becomes closer to $\boldsymbol{d}$, causing $\boldsymbol{\delta}^{(out)}$ to be mostly sparse or close to 0.

Figure B.1 shows some of the results for the FC network. The neural network is fed with

[1] In most practical cases, the input signal's dimension of a fully connected layer is in the order of at least 100s and 1000s, which as verified in our simulations, it is large enough for the validity of our assumptions.

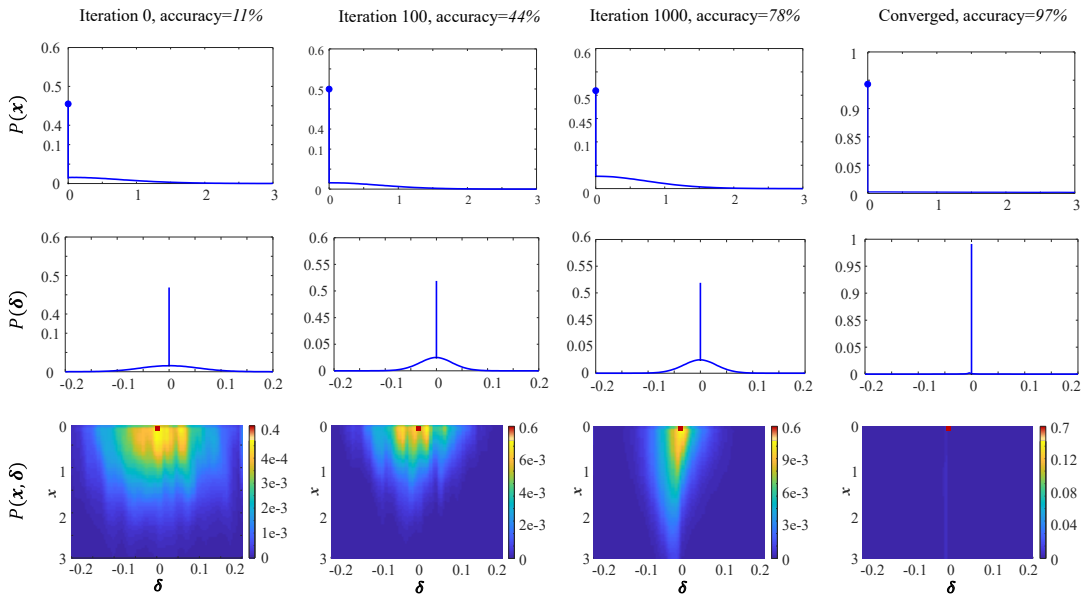


Figure B.1: Marginal distributions of $\boldsymbol{x}^{(1)}$ and $\boldsymbol{\delta}^{(1)}$ in FC network at different stages of training. Note that the y-axis is broken to make the plots legible.

1000s of samples from the database at different stages of training and the probability density functions (pdf) of the signals of different layers are estimated. It is observed that the pdfs of the forward signals closely follow a sparse Folded-Normal distribution and as the neural network trains and its accuracy improves, the sparsity increases. Similarly, the backward signals behave like sparse Gaussian random variables.

This behavior can be explained intuitively as follows. The signals in the hidden layers can be seen as an intermediate "feature vector" derived from input training data and being fed as input to the next layers in the network for classification, decision, ... Hence for different classes or types of inputs from training database, different elements of this "feature vector" would be dominant and the rest become insignificant. This *sparsity* in the signals of the hidden layers becomes more noticeable as the the neural network is being trained and it performance improves.

APPENDIX C

PRACTICAL CONSIDERATIONS IN REPURPOSE

C.1 Complexity of Naive Direct Partitioning

Consider distributing processing of a layer of a deep neural network with N neurons over P workers. Without assuming any constraint on the number of neurons per worker, there are P possible assignments for each neuron, hence, the total possible neuron assignments to the workers would be P^N.

Now, assume that exactly n_k neurons have to be assigned to the k-th worker, where $\sum_k n_k = N$. Clearly, there are

$$\binom{N}{n_1, n_2, \ldots, n_P}$$

possible neuron assignment to the workers. To have a relatively balanced neuron assignment (i.e., no worker or a small subset of workers has to process almost all signals), we assume that $n_k = c_k\, N$, where $c_k = \Theta(1/P)$, i.e., there exists $\alpha, \beta > 0$ such that $\alpha N/P \leq n_k \leq \beta N/P$. Using Stirling's approximation for factorial, $n_k! \sim \sqrt{2\pi n_k}\,(\frac{n_k}{e})^{n_k}$, and noting that $n_k = N\Theta(\frac{1}{P})$, $\sum_k n_k = N$, we have

$$\begin{aligned}
\binom{N}{n_1, n_2, \ldots, n_P} &\sim \frac{\sqrt{2\pi N}\,(\frac{N}{e})^N}{\prod_{k=1}^{P} \sqrt{2\pi n_k}\,(\frac{n_k}{e})^{n_k}} \\
&= \frac{\sqrt{2\pi N}\,(\frac{N}{e})^N}{\prod_{k=1}^{P} \sqrt{2\pi N\Theta(\frac{1}{P})}\,\big(\frac{N\Theta(\frac{1}{P})}{e}\big)^{n_k}} \\
&= \frac{1}{(2\pi N)^{\frac{P}{2}-1}}\,\frac{1}{\Theta(\frac{1}{P^{N+0.5}})} \\
&= \Theta\big(P^{N+0.5}\, N^{1-\frac{P}{2}}\big).
\end{aligned}$$

Therefore, the direct approach to find good neuron assignment for parallel distributed

inference requires evaluation of $\mathcal{O}(P^N)$ different assignments, which for large number of neurons or number of workers becomes prohibitive.

C.2 Reduction in Computational Complexity

One major benefit of applying RePurpose, as demonstrated in simulations, is the reduction in the computational complexity. For the sake of simplicity, assume that there are $P = 2$ workers. Recall that the computations at worker 1 is given as $\boldsymbol{y}_1 = \boldsymbol{W}_{11}^{\mathsf{T}}\boldsymbol{x}_1 + \boldsymbol{b}_1 + \boldsymbol{W}_{12}^{\mathsf{T}}\boldsymbol{x}_2$. By the application of RePurpose to the weight matrix $\boldsymbol{W}$, the off-diagonal blocks, $\boldsymbol{W}_{12}$ and $\boldsymbol{W}_{21}$, become sparse. Let Ω be the indexes of the columns of $\boldsymbol{W}_{12}$ which are non-zero, and define $\widetilde{\boldsymbol{W}}_{12}$ to be the restriction of $\boldsymbol{W}_{12}$ to those non-zero columns. Similarly, define $\widetilde{\boldsymbol{x}}_2$ to be the restriction of $\boldsymbol{x}_2$ to the indexes given by Ω. Therefore, $\boldsymbol{y}_1$ can be more efficiently calculated as $\boldsymbol{y}_1 = \boldsymbol{W}_{11}^{\mathsf{T}}\boldsymbol{x}_1 + \boldsymbol{b}_1 + \widetilde{\boldsymbol{W}}_{12}^{\mathsf{T}}\widetilde{\boldsymbol{x}}_2$. If $\boldsymbol{W}_{12}$ is an $m \times n$ matrix, the computational complexity and the communication requirement of the cross-term $\boldsymbol{W}_{12}^{\mathsf{T}}\boldsymbol{x}_2$ in the original calculation would be $\mathcal{O}(mn)$ and $\mathcal{O}(m)$, respectively. RePurpose reduces these complexities to $\mathcal{O}(|\Omega|n)$ and $\mathcal{O}(|\Omega|)$. As shown in simulations, the set Ω can be extremely small, making the computational complexity of the cross-term negligible. For example, in applying the proposed technique to an $N \times N$ matrix to distributed its computations over 2 workers, if the number of cross dependencies are reduced by a factor of 10, then the computational complexity of matrix multiplication would be reduced to $0.275\, N^2$ per worker, almost 1.8 times reduction from $N^2/2$ in naive parallel implementation.

C.3 Extension of RePurpose to Convolutional Layers

Consider a convolutional layer whose input consists of c_{in} channels of d-dimensional tensors and its output has c_{out} channels. Let $h(z_0, \ldots, z_{d-1}, c_{in}, c_{out})$ be the kernel. For the sake of simplicity in notations, we ignore strides and dilation in convolution operator. Hence, the

output would be

$$O(x_0,\ldots,x_{d-1},k)=\sum_{l=1}^{c_{in}}\sum_{z_0,\ldots,z_{d-1}} h(z_0,\ldots,z_{d-1},l,k)\,I(x_0+z_0,\ldots,x_{d-1}+z_{d-1},l),$$

where $I(\cdot)$ is the input d-dimensional tensor with c_{in} channels and $O(\cdot)$ is the output tensor.

Note that due to the nature of the convolution operator, it is not possible to rearrange the neurons within each channel (e.g., changing locations of pixels in images). However, we propose to change the order of the channels. Note that the convolution can be rewritten as

$$O_k(x_0,\ldots,x_{d-1})=\sum_{l=1}^{c_{in}} h_{l,k} * I_l\,(x_0,\ldots,x_{d-1}),$$

where $h_{l,k}(\cdots)=h(\cdots,l,k)$ is the kernel connecting input channel l to output channel k, $I_l(\cdot)$ is the l-th channel of the input tensor, and $O_k(\cdot)$ is the k-th output channel. Now, similar to (6.4), we can define the cost of assigning *channel* i to the j-th worker as follows:

$$C_{ji}=\min_{\{\widehat{\boldsymbol{h}}_{l,i}\}}\sum_{l=1}^{c_{in}}\|\boldsymbol{h}_{l,i}-\widehat{\boldsymbol{h}}_{l,i}\|_F^2+\eta_1\sum_{l=1}^{c_{in}}\mathbb{I}\Big(\widehat{\boldsymbol{h}}_{l,i}\neq\mathbf{0}\Big)+\eta_2\sum_{l:l\notin\mathcal{C}_{in}(j)}\mathbb{I}\Big(\widehat{\boldsymbol{h}}_{l,i}\neq\mathbf{0}\Big),\qquad\text{(C.1)}$$

where $\mathcal{C}_{in}(j)$ is the set of input channels located at the j-th worker, and $\mathbb{I}(z)=1$ if z is true, and is 0, otherwise. Note that for the convolutional layers, we treat the individual filters as a whole, and the entire channel filter may be set to zero, not the individual coefficients. The solution of (C.1) is given by hard-thresholding,

$$\widehat{\boldsymbol{h}}_{l,i}=\begin{cases}\mathbf{0} & \|\boldsymbol{h}_{l,i}\|_F^2\leq\eta\\ \boldsymbol{h}_{l,i} & \text{o.w.}\end{cases}\qquad\text{(C.2)}$$

where $\eta=\eta_1$ if $l\in\mathcal{C}_{in}(j)$ and $\eta=\eta_1+\eta_2$, otherwise.

With the new assignment cost, RePurpose for convolutional layers is simply given as in Alg. 3.

APPENDIX D

PROOFS

D.1 Proof of Lemma 1

Let $\boldsymbol{e} = \alpha\boldsymbol{g} + \boldsymbol{u} - Q_1(\alpha\boldsymbol{g} + \boldsymbol{u})$ and $\boldsymbol{r} = \boldsymbol{s} - \boldsymbol{u} - \alpha\bar{\tilde{\boldsymbol{g}}}$. Then,

$$\hat{g}_i = \bar{\tilde{g}}_i + \alpha(r_i - Q_2(r_i)).$$

Since $\bar{\tilde{g}}_i = g_i + z_i$, it can be shown that

$$r_i - Q_2(r_i) = \alpha z_i - e_i - Q_2(\alpha z_i - e_i).$$

Therefore,

$$\hat{g}_i = \bar{\tilde{g}}_i + \alpha(\alpha z_i - e_i) - \alpha Q_2(\alpha z_i - e_i).$$

The correct decoding occurs when $Q_2(\alpha z_i - e_i) = 0$. Hence, the probability of correct recovery would be $1 - p$ where

$$p = \Pr\left(|\alpha_z + u| > \frac{\Delta_2}{2}\right), \quad u \sim \mathcal{U}[-\Delta_1/2, \Delta_1/2].$$

In that case,

$$\hat{g}_i = g_i - (\alpha e_i + (1 - \alpha^2) z_i).$$

Since $e_i \sim \mathcal{U}[-\Delta_1/2, \Delta_1/2]$ and z_i are independent from each other and from g_i, simple calculations show that

$$\mathbb{E}\left[(\tilde{g}_i - g_i)^2\right] = \alpha^2 \frac{\Delta_1^2}{12} + (1 - \alpha^2)^2 \sigma_z^2.$$

D.2 Proof of Lemma 3

–First assume that $\boldsymbol{x}$ and $\boldsymbol{\delta}$ are independent random variables.

Unbiasedness. To prove the unbiasedness of the ISGQ, we note that for independent signals $\boldsymbol{x}$ and $\boldsymbol{\delta}$, $\tilde{\boldsymbol{\delta}}$ and $\widetilde{\boldsymbol{x}}$ would be independent as well. Since the individual quantizers are unbiased,

$$\mathbb{E}[\widetilde{\boldsymbol{G}}] = \frac{1}{L}\,\mathbb{E}\big[\widetilde{\boldsymbol{\Delta}}^{\mathsf{T}}\widetilde{\boldsymbol{X}}\big] = \frac{1}{L}\,\mathbb{E}\big[\widetilde{\boldsymbol{\Delta}}^{\mathsf{T}}\big]\,\mathbb{E}[\widetilde{\boldsymbol{X}}] \tag{D.1}$$

$$= \frac{1}{L}\,\mathbb{E}\big[\boldsymbol{\Delta}^{\mathsf{T}}\big]\,\mathbb{E}[\boldsymbol{X}] = \frac{1}{L}\,\mathbb{E}\big[\boldsymbol{\Delta}^{\mathsf{T}}\boldsymbol{X}\big] = \mathbb{E}[\boldsymbol{G}]. \tag{D.2}$$

Therefore, $\widetilde{\boldsymbol{G}}$ is an unbiased stochastic gradient.

Bounded-Variance. Consider an arbitrary element $g = G_{i,j}$ and the corresponding forward and backward signals $\boldsymbol{x} = (\boldsymbol{X}_{j,\cdot})^{\mathsf{T}}$ and $\boldsymbol{\delta} = (\boldsymbol{\Delta}_{i,\cdot})^{\mathsf{T}}$. Recall that the quantizers are designed such that

$$\mathbb{E}\big[\widetilde{\boldsymbol{x}}^{\mathsf{T}}(\boldsymbol{x}-\widetilde{\boldsymbol{x}})\big] = 0. \;\Rightarrow\; \mathbb{E}\big[\|\widetilde{\boldsymbol{x}}\|^2\big] = \mathbb{E}\big[\widetilde{\boldsymbol{x}}^{\mathsf{T}}\boldsymbol{x}\big] \overset{(a)}{\leq} \sqrt{\mathbb{E}[\|\widetilde{\boldsymbol{x}}\|^2]\;\mathbb{E}[\|\boldsymbol{x}\|^2]},$$

where (a) is because of the Cauchy-Schwartz inequality. This implies that

$$\mathbb{E}\big[\|\widetilde{\boldsymbol{x}}\|^2\big] \leq \mathbb{E}\big[\|\boldsymbol{x}\|^2\big]. \tag{D.3}$$

Similarly, $\mathbb{E}\big[\|\widetilde{\boldsymbol{\delta}}\|^2\big] \leq \mathbb{E}[\|\boldsymbol{\delta}\|^2]$. Therefore,

$$\begin{aligned}\mathbb{E}\big[\tilde{g}^2\big] = \frac{1}{L^2}\,\mathbb{E}\Big[(\tilde{\boldsymbol{\delta}}^{\mathsf{T}}\widetilde{\boldsymbol{x}})^2\Big] &\overset{(b)}{\leq} \frac{1}{L^2}\,\mathbb{E}\Big[\|\tilde{\boldsymbol{\delta}}\|^2\;\|\widetilde{\boldsymbol{x}}^2\|^2\Big]\\ &\overset{(c)}{=} \frac{1}{L^2}\,\mathbb{E}\Big[\|\tilde{\boldsymbol{\delta}}\|^2\Big]\;\mathbb{E}\big[\|\widetilde{\boldsymbol{x}}^2\|^2\big]\\ &\overset{(d)}{\leq} \mathbb{E}\big[\|\boldsymbol{\delta}\|^2\big]\;\mathbb{E}\big[\|\boldsymbol{x}\|^2\big] \overset{(e)}{=} \mathbb{E}\big[\|\boldsymbol{\delta}\|^2\;\|\boldsymbol{x}\|^2\big],\end{aligned}$$

where (b) is due to the Cauchy-Schwartz inequality $|\tilde{\boldsymbol{\delta}}^{\mathsf{T}}\widetilde{\boldsymbol{x}}| \leq \|\tilde{\boldsymbol{\delta}}\|\;\|\widetilde{\boldsymbol{x}}^2\|$, and (c) and (e) are

because of the independence of the signals and (d) is due to (D.3). By the assumption, the signals have bounded joint second moment, i.e., $\mathbb{E}[\|\boldsymbol{x}\|^2\|\boldsymbol{\delta}\|^2] < \infty$. Therefore, $\mathbb{E}[\tilde{g}^2]$ is bounded. Since $\tilde{g}$ is unbiased, we conclude that its variance is also bounded.

Performance of One-Bit Indirect Quantizer. Here, we examine properties of 1-bit Naïve ISGQ when $\boldsymbol{x}$ and $\boldsymbol{\delta}$ follow Normal or Folded-Normal distributions. Finding the optimum quantizers is based on verifying the Lloyd-Max optimality conditions of a 1-bit quantizer [117]. For a random variable $u \sim p(u)$, the optimum 1-bit quantizer is given by

$$Q(u) = \begin{cases} c_0 & u \leq \tau \\ c_1 & u > \tau \end{cases},$$

where the optimality conditions imply that $c_0 = \mathbb{E}[u|u \leq \tau]$, $c_1 = \mathbb{E}[u|u > \tau]$ and $\tau = (c_0 + c_1)/2$. E.g., for Normal random variables $\mathcal{N}(0,1)$, $\tau = 0$ and $c_1 = -c_0 = \sqrt{2/\pi}$.

Using the optimality conditions for the individual quantizers of $\boldsymbol{x}$ and $\boldsymbol{\delta}$ for Normal and Folded-Normal distributions and the independence of $\boldsymbol{x}$ and $\boldsymbol{\delta}$, it can be easily verified that

1. If x_k's and δ_k's are i.i.d. Normal random variables, $x_k \sim \mathcal{N}(0, \sigma_x^2)$ and $\delta_k \sim \mathcal{N}(0, \sigma_d^2)$, then $\mathbb{E}[(g - \tilde{g})^2] = \frac{\sigma^2}{L}(1 - \frac{4}{\pi^2})$, where $\sigma = \sigma_x\, \sigma_d$.
2. If x_k's have Folded Normal distribution, $x_k \sim \mathcal{FN}(0, \sigma_x^2)$, and δ_k's are Normal, $\delta_k \sim \mathcal{N}(0, \sigma_d^2)$, then $\mathbb{E}[(g - \tilde{g})^2] = \frac{\sigma^2}{L}(1 - \frac{1.96}{\pi})$, where $\sigma = \sigma_x \sigma_d$.

Simple calculations show that for $L = 1$ the optimum quantizer directly designed for g has the same MSE as the indirect quantizer. However, as $L \to +\infty$, central limit theorem implies that g converges in distribution to a Gaussian random variable with mean 0 and variance σ^2/L. The MSE of the optimum 1-bit quantizer for this Normal random variable is given by $\frac{\sigma^2}{L}(1 - 2/\pi)$. Comparing the MSE of the optimum direct quantizer and the naïve indirect quantizer when x follows a Folded-Normal distribution and δ is Normal, reveals that the difference varies between 0 and 4%.

– Now assume that $\boldsymbol{x}$ and $\boldsymbol{\delta}$ are correlated random variables. In deterministic quantization, it is well-known that the quantization noise is correlated with the input, i.e., $\boldsymbol{e}_x = \boldsymbol{x} - \tilde{\boldsymbol{x}}$

and $\boldsymbol{x}$ are correlated. Since $\boldsymbol{x}$ and $\boldsymbol{\delta}$ are not independent, $\boldsymbol{e}_x$ and $\boldsymbol{\delta}$ are correlated and $\mathbb{E}\left[\boldsymbol{e}_x^{\mathsf{T}}\boldsymbol{\delta}\right]$ would not necessarily vanish. Similar argument shows that $\mathbb{E}[\tilde{g}-g]$ is not zero in general and hence naïve ISGQ for correlated signals is not necessarily unbiased.

As an example, consider correlated normal signals; $x_k \sim \mathcal{N}(0,1)$, $\delta_k \sim \mathcal{N}(0,1)$ and $\mathbb{E}[x_k\delta_k] = \rho$. Hence, the optimum one-bit quantizer for x is given by

$$\tilde{x} = \begin{cases} \sqrt{2/\pi} & x \geq 0 \\ -\sqrt{2/\pi} & x < 0 \end{cases}.$$

Computing Bias. Due to the structure of the individual quantizers, it can be easily verified that

$$\begin{aligned}\mathbb{E}\left[\tilde{\delta}\tilde{x}\right] = \frac{2}{\pi}\big(&\mathrm{P}(x \geq 0,\ \delta \geq 0) + \mathrm{P}(x < 0,\ \delta < 0) \\ &- \mathrm{P}(x < 0,\ \delta \geq 0) - \mathrm{P}(x \geq 0,\ \delta < 0)\big).\end{aligned}$$

Since (x, δ) are jointly Gaussian with correlation ρ, It can be easily verified that

$$\begin{aligned}\mathrm{P}(x \geq 0,\ \delta \geq 0) + \mathrm{P}(x < 0,\ \delta < 0) &= 2\int_0^{+\infty} p(x)\Phi(\frac{\rho x}{\sqrt{1-\rho^2}})\,dx, \\ \mathrm{P}(x < 0,\ \delta \geq 0) + \mathrm{P}(x \geq 0,\ \delta < 0) &= 2\int_0^{+\infty} p(x)\big(1-\Phi(\frac{\rho x}{\sqrt{1-\rho^2}})\big)\,dx,\end{aligned}$$

where $p(x)$ and $\Phi(\cdot)$ are the Normal p.d.f. and c.d.f. of x, respectively. Therefore,

$$\mathbb{E}[\tilde{g}] = \mathbb{E}\left[\tilde{\delta}\tilde{x}\right] = \frac{2}{\pi}\big(4\int_0^{+\infty} p(x)\Phi(\frac{\rho x}{\sqrt{1-\rho^2}})\,dx - 1\big) = \frac{4}{\pi^2}\arcsin(\rho).$$

On the other hand, $\mathbb{E}[g] = \rho$. Therefore,

$$\mathbb{E}[\tilde{g}-g] = \frac{4}{\pi^2}\arcsin(\rho) - \rho.$$

D.3 Proof of Theorem 4

Unbiasedness. First, we note that

$$\mathbb{E}[\widetilde{\boldsymbol{G}}] = \frac{1}{L}\,\mathbb{E}\big[\widetilde{\boldsymbol{\Delta}}\widetilde{\boldsymbol{X}}^\mathsf{T}\big] = \frac{1}{L}\,\mathbb{E}\big[(\boldsymbol{\Delta} - \kappa_\delta \boldsymbol{E}_\delta)(\boldsymbol{X} - \kappa_x \boldsymbol{E}_x)^\mathsf{T}\big],$$

where $\boldsymbol{E}_x = (\boldsymbol{X} - \widetilde{\boldsymbol{X}})/\kappa_x$ and $\boldsymbol{E}_\delta = (\boldsymbol{\Delta} - \widetilde{\boldsymbol{\Delta}})/\kappa_\delta$ are the scaled quantization noises. Since $\mathbb{E}[\boldsymbol{E}_x] = \boldsymbol{0}$, $\mathbb{E}[\boldsymbol{E}_\delta] = \boldsymbol{0}$ and they are independent from other variables,

$$\begin{aligned}\mathbb{E}[\widetilde{\boldsymbol{G}}] = \frac{1}{L}\big(\mathbb{E}\big[\boldsymbol{\Delta}\boldsymbol{X}^\mathsf{T}\big] - \mathbb{E}[\kappa_x\boldsymbol{\Delta}]\,\mathbb{E}\big[\boldsymbol{E}_x^\mathsf{T}\big] - \mathbb{E}[\boldsymbol{E}_\delta]\,\mathbb{E}\big[\kappa_\delta\boldsymbol{X}^\mathsf{T}\big] \\ + \mathbb{E}[\kappa_\delta\kappa_x]\,\mathbb{E}\big[\boldsymbol{E}_\delta\boldsymbol{E}_x^\mathsf{T}\big]\big) = \mathbb{E}[\boldsymbol{G}].\end{aligned}$$

Variance. To compute the variance, we note that

$$\begin{aligned}\widetilde{\boldsymbol{G}} - \nabla_{\boldsymbol{W}}\mathcal{J} &= \widetilde{\boldsymbol{G}} - \boldsymbol{G} + \boldsymbol{G} - \nabla_{\boldsymbol{W}}\mathcal{J} \\ &= \frac{1}{L}\big(\kappa_\delta\kappa_x\boldsymbol{E}_\delta\boldsymbol{E}_x^\mathsf{T} - \kappa_x\boldsymbol{\Delta}\boldsymbol{E}_x^\mathsf{T} - \kappa_\delta\boldsymbol{E}_\delta\boldsymbol{X}^\mathsf{T}\big) + \big(\boldsymbol{G} - \nabla_{\boldsymbol{W}}\mathcal{J}\big).\end{aligned}$$

Since the quantization noises are independent and zero-mean,

$$\begin{aligned}\mathbb{E}\big[\|\widetilde{\boldsymbol{G}} - \nabla_{\boldsymbol{W}}\mathcal{J}\|_F^2\big] = \\ \frac{1}{L^2}\,\mathbb{E}\Big[\big\|\kappa_\delta\kappa_x\boldsymbol{E}_\delta\boldsymbol{E}_x^\mathsf{T} - \kappa_x\boldsymbol{\Delta}\boldsymbol{E}_x^\mathsf{T} - \kappa_\delta\boldsymbol{E}_\delta\boldsymbol{X}^\mathsf{T}\big\|_F^2\Big] + \mathbb{E}\big[\|\boldsymbol{G} - \nabla_{\boldsymbol{W}}\mathcal{J}\|_F^2\big].\end{aligned}$$

Expanding the first term of the RHS and using the fact entries of $\boldsymbol{E}_x$ and $\boldsymbol{E}_\delta$ are uniformly distributed over $(-1/2, 1/2)$, independent of $\boldsymbol{X}$ and $\boldsymbol{\Delta}$, we find that

$$\begin{aligned}\mathbb{E}\big[\|\widetilde{\boldsymbol{G}} - \boldsymbol{G}\|_F^2\big] &= \frac{1}{L^2}\Big[\,\mathbb{E}\big[\|\kappa_\delta\kappa_x\boldsymbol{E}_\delta\boldsymbol{E}_x^\mathsf{T}\|_F^2\big] + \mathbb{E}\big[\|\kappa_x\boldsymbol{\Delta}\boldsymbol{E}_x^\mathsf{T}\|_F^2\big] + \mathbb{E}\big[\|\kappa_\delta\boldsymbol{E}_\delta\boldsymbol{X}^\mathsf{T}\|_F^2\big]\Big] \\ &= \frac{1}{L^2}\Big[\frac{mnL}{12^2}\,\mathbb{E}\big[(\kappa_x\kappa_\delta)^2\big] + \frac{n}{12}\,\mathbb{E}\big[\kappa_x^2\|\boldsymbol{\Delta}\|_F^2\big] + \frac{m}{12}\,\mathbb{E}\big[\kappa_\delta^2\|\boldsymbol{X}\|_F^2\big]\Big] \quad \text{(D.4)}\end{aligned}$$

Using the facts that $\|\boldsymbol{X}\|_F^2 \leq nL\|\boldsymbol{X}\|_\infty^2$, $\kappa_x = \frac{\|\boldsymbol{X}\|_\infty}{K_x}$ and similar points for $\boldsymbol{\Delta}$, we conclude that

$$\begin{aligned}\mathbb{E}\big[\|\widetilde{\boldsymbol{G}} - \boldsymbol{G}\|_F^2\big] \leq \frac{1}{L^2}\Big[\ & \frac{mnL}{12^2}\frac{1}{K_x^2 K_d^2}\,\mathbb{E}\big[\|\boldsymbol{X}\|_\infty^2\|\boldsymbol{\Delta}\|_\infty^2\big] + \\ & \frac{n}{12}\frac{mL}{K_x^2}\,\mathbb{E}\big[\|\boldsymbol{X}\|_\infty^2\|\boldsymbol{\Delta}\|_\infty^2\big] + \frac{m}{12}\frac{nL}{K_d^2}\,\mathbb{E}\big[\|\boldsymbol{X}\|_\infty^2\|\boldsymbol{\Delta}\|_\infty^2\big]\Big] \\ = \frac{mn}{L}\gamma\,\mathbb{E}\big[\|\boldsymbol{X}\|_\infty^2\|\boldsymbol{\Delta}\|_\infty^2\big],\end{aligned}$$

where $\gamma = \frac{1}{144K_x^2K_d^2} + \frac{1}{12K_x^2} + \frac{1}{12K_d^2}$. This completes the proof of the first part of the theorem.

For the special case that $\boldsymbol{X}$ and $\boldsymbol{\Delta}$ are independent Normal random variables, the bound can be improved further by considering the individual terms as follows:

$$\begin{aligned}\mathbb{E}\big[\|\boldsymbol{X}\|_\infty^2\|\boldsymbol{\Delta}\|_\infty^2\big] &= \mathbb{E}\big[\|\boldsymbol{X}\|_\infty^2\big]\,\mathbb{E}\big[\|\boldsymbol{\Delta}\|_\infty^2\big] \\ \mathbb{E}\big[\|\boldsymbol{X}\|_F^2\|\boldsymbol{\Delta}\|_\infty^2\big] &= nL\sigma_x^2\,\mathbb{E}\big[\|\boldsymbol{\Delta}\|_\infty^2\big] \\ \mathbb{E}\big[\|\boldsymbol{X}\|_\infty^2\|\boldsymbol{\Delta}\|_F^2\big] &= mL\sigma_\delta^2\,\mathbb{E}\big[\|\boldsymbol{X}\|_\infty^2\big].\end{aligned}$$

On the other hand,

$$\begin{aligned}\mathbb{E}\big[\|\boldsymbol{X}\|_\infty^2\big] &\leq 4\sigma_x^2\ln\big(\sqrt{2}nL\big), \\ \mathbb{E}\big[\|\boldsymbol{\Delta}\|_\infty^2\big] &\leq 4\sigma_\delta^2\ln\big(\sqrt{2}mL\big).\end{aligned}$$

By substituting the terms, we obtain the desired result.

D.4 Proof of Theorem 5

First, we note that as a result of Hölder inequality, for all $0 < r < s$

$$\mathbb{E}[\|\boldsymbol{X}\|^r] \leq \big(\mathbb{E}[\|\boldsymbol{X}\|^s]\big)^{r/s},$$

which in conjunction with assumption **A3** implies that the moments $\mathbb{E}\left[\|\widetilde{\boldsymbol{G}} - \boldsymbol{G}\|_F^k\right]$ are bounded for $k \leq 4$. Further, note that using Cauchy-Schwartz inequality, we conclude that $\mathbb{E}\left[\|\boldsymbol{G}\|_F^k\right]$ is also bounded for $k \leq 4$. It can be easily verified that if the above assumptions are satisfied, then the conditions of [98, §5.1] are satisfied and the learning algorithm converges to a local extremum almost surely.

D.5 Proof of Lemma 6

Recall that $\boldsymbol{T} = \frac{1}{\sqrt{k}}\boldsymbol{H}\boldsymbol{R}$. Hence,

$$\boldsymbol{T}\boldsymbol{T}^{\mathsf{T}} = \frac{1}{k}\boldsymbol{H}\boldsymbol{R}\boldsymbol{R}^{\mathsf{T}}\boldsymbol{H}^{\mathsf{T}} \overset{(a)}{=} \frac{1}{k}\boldsymbol{H}\boldsymbol{H}^{\mathsf{T}} \overset{(b)}{=} \frac{n}{k}\boldsymbol{I},$$

where (a) is due to the fact that $\boldsymbol{R} = \mathrm{diag}(\boldsymbol{r})$ and $r_i^2 = 1$, and (b) is a result of $\boldsymbol{H}$ being any k rows of Hadamard matrix $\boldsymbol{H}_n$ satisfying $\boldsymbol{H}_n\boldsymbol{H}_n^{\mathsf{T}} = n\boldsymbol{I}$.

For the second property,

$$\mathbb{E}\left[\boldsymbol{T}^{\mathsf{T}}\boldsymbol{T}\right] = \frac{1}{k}\,\mathbb{E}\left[\boldsymbol{R}\boldsymbol{H}\boldsymbol{H}^{\mathsf{T}}\boldsymbol{R}^{\mathsf{T}}\right].$$

Now, consider an arbitrary (i, j)-th element,

$$[\boldsymbol{R}^{\mathsf{T}}\boldsymbol{H}^{\mathsf{T}}\boldsymbol{H}\boldsymbol{R}]_{i,j} = r_i\, r_j\ \left[\boldsymbol{H}^{\mathsf{T}}\boldsymbol{H}\right]_{i,j}.$$

On the other hand, $\mathbb{E}[r_i\, r_j] = 1$ if $i = j$ and 0 if $i \neq j$. Moreover, since $H_{i,l} = \pm 1$, $\left[\boldsymbol{H}^{\mathsf{T}}\boldsymbol{H}\right]_{i,i} = \sum_{l=1}^{k}\left(\boldsymbol{H}_{i,l}\right)^2 = k$. Therefore,

$$\mathbb{E}\left[[\boldsymbol{R}^{\mathsf{T}}\boldsymbol{H}^{\mathsf{T}}\boldsymbol{H}\boldsymbol{R}]_{i,j}\right] = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases}$$

D.6 Proof of Theorem 7

For a fixed $\boldsymbol{g}$, we note that the randomness in $\widehat{\boldsymbol{g}}$ stems from the random mixing matrix $\boldsymbol{T}$ and dither signal $\boldsymbol{u}$, which are independent of each other and $\boldsymbol{g}$. Moreover, the quantization noise of $\boldsymbol{v}$ can be written as

$$\widehat{\boldsymbol{v}} = \boldsymbol{v} - \varrho\boldsymbol{\varepsilon} = \boldsymbol{T}\boldsymbol{g} - \varrho\boldsymbol{\varepsilon}, \tag{D.5}$$

where as a result of Thm. 18, $\boldsymbol{\varepsilon}$ is an independent random variable and $\boldsymbol{\varepsilon} \sim \mathcal{U}(-1/2, 1/2)$.[1] Therefore, the quantization noise can be decomposed as

$$\boldsymbol{e} = \boldsymbol{g} - \widehat{\boldsymbol{g}} = \overbrace{(\boldsymbol{I} - \boldsymbol{T}^{\mathsf{T}}\boldsymbol{T})\boldsymbol{g}}^{\boldsymbol{e}_g} + \overbrace{\varrho\boldsymbol{T}^{\mathsf{T}}\boldsymbol{\varepsilon}}^{\boldsymbol{e}_d}. \tag{D.6}$$

Unbiasedness.

$$\mathbb{E}[\boldsymbol{e}_g] = (\boldsymbol{I} - \mathbb{E}\left[\boldsymbol{T}^{\mathsf{T}}\boldsymbol{T}\right])\boldsymbol{g} \overset{(a)}{=} \boldsymbol{0},$$
$$\mathbb{E}[\boldsymbol{e}_d] = \mathbb{E}\left[\varrho\boldsymbol{T}^{\mathsf{T}}\boldsymbol{\varepsilon}\right] \overset{(b)}{=} \mathbb{E}\left[\varrho\boldsymbol{T}^{\mathsf{T}}\right]\mathbb{E}[\boldsymbol{\varepsilon}] = \boldsymbol{0},$$

where (a) is due to $\mathbb{E}\left[\boldsymbol{T}^{\mathsf{T}}\boldsymbol{T}\right] = \boldsymbol{I}$ (Lemma 6) and (b) is because of independence of $\boldsymbol{\varepsilon}$ from $\varrho\boldsymbol{T}$ and $\mathbb{E}[\boldsymbol{\varepsilon}] = \boldsymbol{0}$. This proves the unbiasedness of QCS.

Variance. Note that since $\boldsymbol{e}_g$ is a function of only $\boldsymbol{T}$ and $\boldsymbol{g}$, it is independent of $\boldsymbol{\varepsilon}$ and

$$\mathbb{E}\left[\boldsymbol{e}_g^{\mathsf{T}}\boldsymbol{e}_d\right] = \mathbb{E}\left[\varrho\boldsymbol{e}_g^{\mathsf{T}}\boldsymbol{T}^{\mathsf{T}}\right]\mathbb{E}[\boldsymbol{\varepsilon}] = 0.$$

Therefore,

$$\mathbb{E}\left[\|\boldsymbol{e}\|_2^2\right] = \mathbb{E}\left[\|\boldsymbol{e}_g\|_2^2\right] + \mathbb{E}\left[\|\boldsymbol{e}_d\|_2^2\right]$$

[1] Note that this is not the case for ordinary quantization or stochastic quantization of [15].

For an arbitrary $\boldsymbol{g}$, note that

$$\mathbb{E}\big[\|\boldsymbol{T}\boldsymbol{g}\|_2^2\big] = \boldsymbol{g}^\mathsf{T}\,\mathbb{E}\big[\boldsymbol{T}^\mathsf{T}\boldsymbol{T}\big]\boldsymbol{g} = \boldsymbol{g}^\mathsf{T}\boldsymbol{I}\boldsymbol{g} = \|\boldsymbol{g}\|_2^2$$
$$\mathbb{E}\big[\|\boldsymbol{T}^\mathsf{T}\boldsymbol{T}\boldsymbol{g}\|_2^2\big] = \mathbb{E}\big[\boldsymbol{g}^\mathsf{T}\boldsymbol{T}^\mathsf{T}\boldsymbol{T}\boldsymbol{T}^\mathsf{T}\boldsymbol{T}\boldsymbol{g}\big] = \frac{n}{k}\,\mathbb{E}\big[\boldsymbol{g}^\mathsf{T}\boldsymbol{T}^\mathsf{T}\boldsymbol{T}\boldsymbol{g}\big] = \frac{n}{k}\|\boldsymbol{g}\|_2^2.$$

Therefore,

$$\begin{aligned}\mathbb{E}\big[\|\boldsymbol{e}_g\|_2^2\big] &= \mathbb{E}\big[\|(\boldsymbol{I}-\boldsymbol{T}^\mathsf{T}\boldsymbol{T})\boldsymbol{g}\|_2^2\big]\\ &= \mathbb{E}\big[\|\boldsymbol{T}^\mathsf{T}\boldsymbol{T}\boldsymbol{g}\|_2^2\big] + \|\boldsymbol{g}\|_2^2 - 2\,\mathbb{E}\big[\|\boldsymbol{T}\boldsymbol{g}\|_2^2\big] = (\frac{n}{k}-1)\|\boldsymbol{g}\|_2^2.\end{aligned}$$

On the other hand,

$$\begin{aligned}\mathbb{E}\big[\|\boldsymbol{e}_d\|_2^2\big] &= \mathbb{E}\big[\|\varrho\boldsymbol{T}^\mathsf{T}\boldsymbol{\varepsilon}\|_2^2\big] = \mathbb{E}_{\boldsymbol{T}}\big[\mathbb{E}_{\boldsymbol{\varepsilon}}\big[\|\varrho\boldsymbol{T}^\mathsf{T}\boldsymbol{\varepsilon}\|_2^2|\boldsymbol{T}\big]\big]\\ &= \mathbb{E}_{\boldsymbol{T}}\big[\varrho^2\,\mathbb{E}_{\boldsymbol{\varepsilon}}\big[\|\boldsymbol{T}^\mathsf{T}\boldsymbol{\varepsilon}\|_2^2|\boldsymbol{T}\big]\big] \overset{(c)}{=} \mathbb{E}_{\boldsymbol{T}}\left[\varrho^2\frac{\|\boldsymbol{T}\|_F^2}{12}\right] \overset{(d)}{=} \frac{n}{12Q^2}\,\mathbb{E}_{\boldsymbol{T}}\big[\|\boldsymbol{T}\boldsymbol{g}\|_\infty^2\big],\end{aligned}$$

where (c) is due to $\boldsymbol{\varepsilon}$ being i.i.d. $\mathcal{U}(-1/2, 1/2)$ and (d) is because of $\|\boldsymbol{T}\|_F^2 = n$ and definition of $\varrho = \|\boldsymbol{T}\boldsymbol{g}\|_\infty/Q$.

To bound $\mathbb{E}_{\boldsymbol{T}}[\|\boldsymbol{T}\boldsymbol{g}\|_\infty^2]$ we need the following lemma.

Lemma 20. *Let $\boldsymbol{a} \in \mathbb{R}^n$ be fixed and $\boldsymbol{r}$ be an i.i.d. Rademacher random vector. Then for all $0 \le \lambda < 1/(2\|\boldsymbol{a}\|_2^2)$,*

$$\mathbb{E}_{\boldsymbol{r}}\Big[e^{\lambda(\boldsymbol{a}^\mathsf{T}\boldsymbol{r})^2}\Big] \le \frac{1}{\sqrt{1-2\lambda\|\boldsymbol{a}\|_2^2}} \tag{D.7}$$

Proof. Let $\omega \sim \mathcal{N}(0,1)$ be an independent normal random variable. Note that for a fixed $\boldsymbol{r}$,

$$e^{\lambda(\boldsymbol{a}^\mathsf{T}\boldsymbol{r})^2} = \mathbb{E}_\omega\Big[\exp\Big(\sqrt{2\lambda}(\boldsymbol{a}^\mathsf{T}\boldsymbol{r})\omega\Big)\Big].$$

Therefore,

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{r}}\left[e^{\lambda(\boldsymbol{a}^\mathsf{T}\boldsymbol{r})^2}\right] &= \mathbb{E}_{\boldsymbol{r}}\left[\mathbb{E}_{\omega}\left[\exp\left(\sqrt{2\lambda}(\boldsymbol{a}^\mathsf{T}\boldsymbol{r})\omega\right)|\boldsymbol{r}\right]\right] \\
&= \mathbb{E}_{\omega}\left[\mathbb{E}_{\boldsymbol{r}}\left[\exp\left(\sqrt{2\lambda}(\boldsymbol{a}^\mathsf{T}\boldsymbol{r})\omega\right)|\omega\right]\right] \\
&\overset{(e)}{=} \mathbb{E}_{\omega}\left[\prod_{i=1}^{n}\mathbb{E}_{r_i}\left[\exp\left(\sqrt{2\lambda}\omega a_i r_i\right)|\omega\right]\right] \\
&\overset{(f)}{\leq} \mathbb{E}_{\omega}\left[\prod_{i=1}^{n}\exp\left(\lambda\omega^2 a_i^2\right)\right] \\
&= \mathbb{E}_{\omega}\left[\exp\left(\lambda\|\boldsymbol{a}\|_2^2\omega^2\right)\right] = \frac{1}{\sqrt{1-2\lambda\|\boldsymbol{a}\|_2^2}}
\end{aligned}$$

where (e) is because of independence of r_i's, (f) is from Hoeffding's lemma, ■

Using the above lemma, for $\boldsymbol{v} = \boldsymbol{T}\boldsymbol{g}$ and arbitrary $\lambda > 0$,

$$\begin{aligned}
\mathbb{E}\left[\|\boldsymbol{v}\|_\infty^2\right] &\leq \frac{1}{\lambda}\,\mathbb{E}\left[\log\left(\sum_{i=1}^{k}\exp\left(\lambda v_i^2\right)\right)\right] \\
&\leq \frac{1}{\lambda}\log\mathbb{E}\left[\sum_{i=1}^{k}\exp\left(\lambda v_i^2\right)\right] \\
&= \frac{1}{\lambda}\log\left(\sum_{i=1}^{k}\mathbb{E}\left[\exp(\lambda v_i^2)\right]\right)
\end{aligned}$$

Note that for an arbitrary fixed i, $\zeta_j := r_j H_{i,j}$ would be i.i.d. Rademacher random variables and $v_i = \sum_j (\frac{g_j}{\sqrt{k}})\zeta_j$. Therefore, by Lemma 20

$$\mathbb{E}\left[\exp(\lambda v_i^2)\right] \leq \frac{1}{\sqrt{1-2\lambda\|\boldsymbol{g}\|_2^2/k}},$$

and

$$\mathbb{E}\left[\|\boldsymbol{v}\|_\infty^2\right] \leq \frac{1}{\lambda}\log\left(\frac{k}{\sqrt{1-2\lambda\|\boldsymbol{g}\|_2^2/k}}\right).$$

For $k \geq 2$, let $\lambda = \frac{k}{2\|\boldsymbol{g}\|_2^2}(1-k^{-\delta})$. Therefore,

$$\frac{1}{\lambda}\log\left(\frac{k}{\sqrt{1-2\lambda\|\boldsymbol{g}\|_2^2/k}}\right) = \frac{2\|\boldsymbol{g}\|_2^2}{k}\,\frac{1}{1-k^{-\delta}}\log(k^{1+\delta/2})$$
$$= \|\boldsymbol{g}\|_2^2\,\frac{\log(k)}{k}\,\frac{2+\delta}{1-k^{-\delta}}. \tag{D.8}$$

Setting $\delta = 1$, results in the bound

$$\mathbb{E}\big[\|\boldsymbol{v}\|_\infty^2\big] \leq 3\|\boldsymbol{g}\|_2^2\,\frac{\log(k)}{k-1}. \tag{D.9}$$

Summarizing the above results for $k \geq 2$, we have

$$\mathbb{E}\big[\|\widehat{\boldsymbol{g}}-\boldsymbol{g}\|_2^2\big] \leq \left(\frac{n}{k}-1+\frac{n}{4Q^2}\,\frac{\log(k)}{k-1}\right)\|\boldsymbol{g}\|_2^2. \tag{D.10}$$

For $k = 1$, note that since v is a scalar, by the definition of the used dithered quantizer sending magnitude of v and its sign results in $\hat{v} = v$ and $e_d = 0$. Therefore,

$$\mathbb{E}\big[\|\widehat{\boldsymbol{g}}-\boldsymbol{g}\|_2^2\big] = (n-1)\|\boldsymbol{g}\|_2^2. \tag{D.11}$$

D.7 Proof of Lemma 8

For $\widehat{\boldsymbol{g}} = \alpha\boldsymbol{T}^\mathsf{T}\widehat{\boldsymbol{v}}$, using the same argument as for the unbiased QCS, it can be easily shown that

$$\mathbb{E}\big[\|\boldsymbol{g}-\widehat{\boldsymbol{g}}\|_2^2\big] = \mathbb{E}\big[\|\boldsymbol{g}-\alpha\boldsymbol{T}^\mathsf{T}\boldsymbol{T}\boldsymbol{g}+\alpha\varrho\boldsymbol{T}^\mathsf{T}\boldsymbol{\varepsilon}\|_2^2\big]$$
$$= \|\boldsymbol{g}\|_2^2(1-2\alpha+\alpha^2\frac{n}{k}) + \alpha^2\frac{n}{12Q^2}\,\mathbb{E}\big[\|\boldsymbol{T}\boldsymbol{g}\|_\infty^2\big]$$
$$\leq \|\boldsymbol{g}\|_2^2\big(1-2\alpha+\alpha^2\frac{n}{k}+\alpha^2\frac{n}{4Q^2}\,\frac{\log(k)}{k-1}\big)$$
$$= \|\boldsymbol{g}\|_2^2\big(1-2\alpha+\alpha^2(\gamma+1)\big),$$

where $\gamma = \frac{n}{k} + \frac{n}{4Q^2}\,\frac{\log(k)}{k-1}$ for $k \geq 2$, and $\gamma = n-1$ for $k = 1$.

Minimizing the upper bound of the error results in

$$\alpha = \frac{1}{1+\gamma},$$

and by substituting α, the minimum mean squared error is given by

$$\mathbb{E}\left[\|\boldsymbol{g} - \widehat{\boldsymbol{g}}\|_2^2\right] \leq \|\boldsymbol{g}\|_2^2 \frac{\gamma}{1+\gamma}$$

D.8 Proof of Lemma 9

Recall that $\boldsymbol{e}_t = \boldsymbol{z}_t - \widehat{\boldsymbol{z}}_t = (\boldsymbol{I} - \alpha \boldsymbol{T}_t^{\mathsf{T}} \boldsymbol{T}_t)\boldsymbol{z}_t - \alpha \varrho \boldsymbol{T}_t^{\mathsf{T}} \boldsymbol{\varepsilon}$, where $\alpha = 1$ for Unbiased-QCS and $\alpha = 1/(1+\gamma)$ for MMSE-QCS, and $\boldsymbol{\varepsilon}$ is the scaled quantization noise of $\boldsymbol{T}\boldsymbol{z}_t$. Therefore,

$$\begin{aligned}\mathbb{E}\left[\|\boldsymbol{r}_{t+1}\|_2^2\right] &= (1-\beta)^2\,\mathbb{E}\left[\|\boldsymbol{r}_t\|_2^2\right] + \mathbb{E}\left[\|\boldsymbol{e}_t\|_2^2\right] \\ &\quad + 2(1-\beta)\,\mathbb{E}\left[\boldsymbol{r}_t^{\mathsf{T}} \boldsymbol{e}_t\right].\end{aligned}$$

On the other hand,

$$\begin{aligned}\mathbb{E}\left[\boldsymbol{r}_t^{\mathsf{T}} \boldsymbol{e}_t\right] &= \mathbb{E}\left[\boldsymbol{r}_t^{\mathsf{T}} (\boldsymbol{I} - \alpha \boldsymbol{T}_t^{\mathsf{T}} \boldsymbol{T}_t)\boldsymbol{z}_t\right] - \alpha\,\mathbb{E}\left[\boldsymbol{r}_t^{\mathsf{T}} \varrho \boldsymbol{T}_t^{\mathsf{T}} \boldsymbol{\varepsilon}\right] \\ &= \mathbb{E}\left[\mathbb{E}_{\boldsymbol{T}}\left[\boldsymbol{r}_t^{\mathsf{T}} (\boldsymbol{I} - \alpha \boldsymbol{T}_t^{\mathsf{T}} \boldsymbol{T}_t)\boldsymbol{z}_t | \boldsymbol{r}_t, \cdots\right]\right] \\ &\quad - \alpha\,\mathbb{E}\left[\mathbb{E}_{\boldsymbol{\varepsilon}}\left[\boldsymbol{r}_t^{\mathsf{T}} \varrho \boldsymbol{T}_t^{\mathsf{T}} \boldsymbol{\varepsilon} | \boldsymbol{T}_t, \boldsymbol{r}_t, \cdots\right]\right] \\ &\overset{(a)}{=} \mathbb{E}\left[\mathbb{E}_{\boldsymbol{T}}\left[\boldsymbol{r}_t^{\mathsf{T}} (\boldsymbol{I} - \alpha \boldsymbol{T}_t^{\mathsf{T}} \boldsymbol{T}_t)\boldsymbol{z}_t | \boldsymbol{r}_t, \cdots\right]\right] \\ &\overset{(b)}{=} (1-\alpha)\,\mathbb{E}\left[\boldsymbol{r}_t^{\mathsf{T}} \boldsymbol{z}_t\right],\end{aligned}$$

where (a) is because of $\boldsymbol{\varepsilon}$ being an independent zero-mean random vector and (b) due to $\mathbb{E}_{\boldsymbol{T}}\left[\boldsymbol{T}^{\mathsf{T}} \boldsymbol{T}\right] = \boldsymbol{I}$.

Therefore,

$$\mathbb{E}\big[\|\boldsymbol{r}_{t+1}\|_2^2\big] = (1-\beta)^2\,\mathbb{E}\big[\|\boldsymbol{r}_t\|_2^2\big] + \mathbb{E}\big[\|\boldsymbol{e}_t\|_2^2\big] + 2(1-\beta)(1-\alpha)\,\mathbb{E}\big[\boldsymbol{r}_t^{\mathsf{T}}\boldsymbol{z}_t\big].$$

First, we consider the Unbiased-QCS.

Lemma 21. *In Unbiased-QCS with error-feedback, residual signal and the stochastic gradients are uncorrelated, i.e.,* $\forall t, \tau$*:* $\mathbb{E}\big[\boldsymbol{g}_t^{\mathsf{T}}\boldsymbol{r}_\tau\big] = 0$*.*

Proof. The proof is based on induction. For $\tau = 0$, since $\boldsymbol{r}_\tau = \mathbf{0}$, the claim holds. Assume that the claim is true for $\tau - 1$.

$$\begin{aligned}
\mathbb{E}\big[\boldsymbol{g}_t^{\mathsf{T}}\boldsymbol{r}_\tau\big] &= \mathbb{E}\big[\boldsymbol{g}_t^{\mathsf{T}}((1-\beta)\boldsymbol{r}_{\tau-1} + \boldsymbol{e}_\tau)\big] \\
&= (1-\beta)\,\mathbb{E}\big[\boldsymbol{g}_t^{\mathsf{T}}\boldsymbol{r}_{\tau-1}\big] + \mathbb{E}\big[\boldsymbol{g}_t^{\mathsf{T}}\boldsymbol{e}_\tau\big] \\
&= \mathbb{E}\big[\boldsymbol{g}_t^{\mathsf{T}}((\boldsymbol{I} - \boldsymbol{T}_\tau^{\mathsf{T}}\boldsymbol{T}_\tau)\boldsymbol{z}_\tau + \varrho\boldsymbol{T}_\tau^{\mathsf{T}}\boldsymbol{\varepsilon}_\tau)\big] \\
&= \mathbb{E}\big[\mathbb{E}_{\boldsymbol{T}_\tau}\big[\boldsymbol{g}_t^{\mathsf{T}}(\boldsymbol{I} - \boldsymbol{T}_\tau^{\mathsf{T}}\boldsymbol{T}_\tau)\boldsymbol{z}_\tau\big|\boldsymbol{g}_t, \boldsymbol{z}_\tau\big]\big] + \\
&\quad \mathbb{E}\big[\mathbb{E}_{\boldsymbol{\varepsilon}_\tau}\big[\varrho\boldsymbol{g}_t^{\mathsf{T}}\boldsymbol{T}_\tau^{\mathsf{T}}\boldsymbol{\varepsilon}_\tau\big|\boldsymbol{g}_t, \boldsymbol{T}_{\boldsymbol{\tau}}\big]\big] = 0.
\end{aligned}$$

■

Since in unbiased QCS, $\alpha = 1$, we have

$$\begin{aligned}
\mathbb{E}\big[\|\boldsymbol{r}_{t+1}\|_2^2\big] &= (1-\beta)^2\,\mathbb{E}\big[\|\boldsymbol{r}_t\|_2^2\big] + \mathbb{E}\big[\|\boldsymbol{e}_t\|_2^2\big] \\
&\overset{(c)}{\leq} (1-\beta)^2\,\mathbb{E}\big[\|\boldsymbol{r}_t\|_2^2\big] + \gamma\,\mathbb{E}\big[\|\boldsymbol{z}_t\|_2^2\big] \\
&= (1-\beta)^2\,\mathbb{E}\big[\|\boldsymbol{r}_t\|_2^2\big] + \gamma\big(\mathbb{E}\big[\|\boldsymbol{g}_t + \beta\boldsymbol{r}_t\|_2^2\big]\big) \\
&\overset{(d)}{=} \big((1-\beta)^2 + \beta^2\gamma\big)\,\mathbb{E}\big[\|\boldsymbol{r}_t\|_2^2\big] + \gamma\,\mathbb{E}\big[\|\boldsymbol{g}_t\|_2^2\big] \\
&\overset{(e)}{\leq} \big((1-\beta)^2 + \beta^2\gamma\big)\,\mathbb{E}\big[\|\boldsymbol{r}_t\|_2^2\big] + \gamma B
\end{aligned} \tag{D.12}$$

where (c) is due to the fact that for all $\boldsymbol{z}$, $\mathbb{E}[\|\boldsymbol{z}-\widehat{\boldsymbol{z}}\|_2^2] \leq \gamma\|\boldsymbol{z}\|_2^2$, (d) is from Lemma 21 and (e) is from boundedness of $\boldsymbol{g}$. The recursive equation (D.12) with $\boldsymbol{r}_0 = \mathbf{0}$ implies that

$$\mathbb{E}\big[\|\boldsymbol{r}_t\|_2^2\big] \leq \frac{\gamma}{1-\big((1-\beta)^2+\beta^2\gamma\big)}B, \tag{D.13}$$

for all β that $\beta < 1$ and $(1-\beta)^2+\beta^2\gamma < 1$, hence, $\beta < \min(1, 2/(1+\gamma))$.

For MMSE-QCS, the stochastic gradients and residual signal might be correlated. However, for an arbitrary $c > 0$, their correlation can be bounded as

$$\begin{aligned}
0 \leq \mathbb{E}\left[\|\frac{1}{\sqrt{c}}\boldsymbol{g}_t \pm \sqrt{c}\boldsymbol{r}_\tau\|_2^2\right] &= \frac{1}{c}\,\mathbb{E}\big[\|\boldsymbol{g}_t\|_2^2\big] + c\,\mathbb{E}\big[\|\boldsymbol{r}_\tau\|_2^2\big] \\
&\quad \pm 2\,\mathbb{E}\big[\boldsymbol{g}_t^{\mathsf{T}}\boldsymbol{r}_\tau\big] \\
\Rightarrow 2\big|\,\mathbb{E}\big[\boldsymbol{g}_t^{\mathsf{T}}\boldsymbol{r}_\tau\big]\,\big| &\leq \frac{1}{c}\,\mathbb{E}\big[\|\boldsymbol{g}_t\|_2^2\big] + c\,\mathbb{E}\big[\|\boldsymbol{r}_\tau\|_2^2\big].
\end{aligned}$$

Therefore, noting that $\mathbb{E}[\|\boldsymbol{e}\|_2^2] \leq (1-\alpha)\|\boldsymbol{z}\|_2^2$ and $\mathbb{E}[\|\boldsymbol{g}_t\|_2^2] \leq B$,

$$\begin{aligned}
\mathbb{E}\big[\|\boldsymbol{r}_{t+1}\|_2^2\big] &= (1-\beta)^2\,\mathbb{E}\big[\|\boldsymbol{r}_t\|_2^2\big] + \mathbb{E}\big[\|\boldsymbol{e}_t\|_2^2\big] \\
&\quad + 2(1-\beta)(1-\alpha)\,\mathbb{E}\big[\boldsymbol{r}_t^{\mathsf{T}}\boldsymbol{z}_t\big] \\
&\leq (1-\beta)^2\,\mathbb{E}\big[\|\boldsymbol{r}_t\|_2^2\big] + (1-\alpha)\,\mathbb{E}\big[\|\boldsymbol{g}_t+\beta\boldsymbol{r}_t\|_2^2\big] + \\
&\quad 2(1-\beta)(1-\alpha)\,\mathbb{E}\big[\boldsymbol{r}_t^{\mathsf{T}}(\boldsymbol{g}_t+\beta\boldsymbol{r}_t)\big] \\
&\leq \big(1-\alpha\beta(2-\beta)\big)\,\mathbb{E}\big[\|\boldsymbol{r}_t\|_2^2\big] + \\
&\quad (1-\alpha)(c\,\mathbb{E}\big[\|\boldsymbol{r}_t\|_2^2\big] + \frac{1}{c}B) + (1-\alpha)B \\
&= \big(1-\alpha\beta(2-\beta)+c(1-\alpha)\big)\,\mathbb{E}\big[\|\boldsymbol{r}_t\|_2^2\big] + \\
&\quad (1-\alpha)(1+\frac{1}{c})B.
\end{aligned}$$

Therefore, if $\left|1-\alpha\beta(2-\beta)+c(1-\alpha)\right|<1$,

$$\mathbb{E}\left[\|\boldsymbol{r}_t\|_2^2\right] \leq \frac{1+1/c}{\alpha\beta(2-\beta)-c(1-\alpha)}(1-\alpha)B. \tag{D.14}$$

Minimizing w.r.t. c and substituting $\alpha = 1/(1+\gamma)$ results in

$$\mathbb{E}\left[\|\boldsymbol{r}_t\|_2^2\right] \leq \frac{1-\alpha}{\left(\sqrt{1-\alpha(1-\beta)^2}-\sqrt{1-\alpha}\right)^2}B \tag{D.15}$$

$$= \frac{\gamma}{\left(\sqrt{\gamma+1-(1-\beta)^2}-\sqrt{\gamma}\right)^2}B. \tag{D.16}$$

Note that if the SGs have bounded variance, i.e., $\mathbb{E}[\|\boldsymbol{g}-\nabla f\|_2^2] \leq \sigma^2$, a similar approach can be used to bound $\mathbb{E}[\|\boldsymbol{r}_t\|_2^2]$ based on the σ^2 and the weighted average of $\|\nabla f(\boldsymbol{w}_{t-i})\|_2^2$ for $i=0,\ldots,t$. This is specially helpful when analyzing the convergence of the training algorithm with error feedback under the assumption of bounded variance SG.

D.9 Proof of Lemma 10

The proof follows the same line of argument as for ordinary SGD which is repeated here for the sake of completeness.

Recall that for Lipschitz-smooth function $f(\cdot)$, for arbitrary $\boldsymbol{w}$ and $\boldsymbol{\delta}$,

$$f(\boldsymbol{w}+\boldsymbol{\delta}) \leq f(\boldsymbol{w}) + \boldsymbol{\delta}^{\mathsf{T}}\nabla f(\boldsymbol{w}) + \frac{L}{2}\|\boldsymbol{\delta}\|_2^2.$$

First, we consider Unbiased-QCS. Tt the t-th iteration of training, $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \mu\widehat{\boldsymbol{g}}_t$, where $\mathbb{E}[\widehat{\boldsymbol{g}}_t] = \nabla f(\boldsymbol{w}_t)$ and $\mathbb{E}[\|\widehat{\boldsymbol{g}}_t\|_2^2] \leq (1+\gamma)\,\mathbb{E}[\|\boldsymbol{g}\|_2^2] \leq (1+\gamma)B$. Hence,

$$\begin{aligned}\mathbb{E}[f(\boldsymbol{w}_{t+1})] &\leq f(\boldsymbol{w}_t) + \langle\nabla f(\boldsymbol{w}_t), \mathbb{E}[\boldsymbol{w}_{t+1}-\boldsymbol{w}_t]\rangle \\ &\quad + \frac{L}{2}\,\mathbb{E}\left[\|\boldsymbol{w}_{t+1}-\boldsymbol{w}_t\|_2^2\right] \\ &\leq f(\boldsymbol{w}_t) - \mu\|\nabla f(\boldsymbol{w}_t)\|_2^2 + \frac{L}{2}\mu^2(1+\gamma)B.\end{aligned}$$

Rearranging terms, taking expectation and summing from $t = 0$ to $T - 1$, results in

$$\begin{aligned}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\big[\|\nabla f(\boldsymbol{w}_t)\|_2^2\big] &\leq \frac{f(\boldsymbol{w}_0) - \mathbb{E}[f(\boldsymbol{w}_T)]}{T\mu} + \frac{L}{2}\mu(1+\gamma)B \\ &\leq \frac{f(\boldsymbol{w}_0) - f^*}{T\mu} + \frac{L}{2}\mu(1+\gamma)B.\end{aligned}$$

Setting $\mu = 1/\sqrt{T}$, results in

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\big[\|\nabla f(\boldsymbol{w}_t)\|_2^2\big] \leq \frac{f(\boldsymbol{w}_0) - f^* \ + \ \frac{L}{2}(1+\gamma)B}{\sqrt{T}}.$$

Note that in the case that the stochastic gradients have bounded variance[2], i.e., $\mathbb{E}[\|\boldsymbol{g} - \nabla f\|_2^2] \leq \sigma^2$, $\mathbb{E}[\|\widehat{\boldsymbol{g}}_t\|_2^2] \leq (1+\gamma)\,\mathbb{E}[\|\boldsymbol{g}\|_2^2] \leq (1+\gamma)(\sigma^2 + \|\nabla f\|_2^2)$ and we can modify the above argument as follows to bound the convergence rate,

$$\begin{aligned}\mathbb{E}[f(\boldsymbol{w}_{t+1})] - f(\boldsymbol{w}_t) &\leq + \langle \nabla f(\boldsymbol{w}_t), \mathbb{E}[\boldsymbol{w}_{t+1} - \boldsymbol{w}_t]\rangle \\ &\quad + \frac{L}{2}\,\mathbb{E}\big[\|\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\|_2^2\big] \\ &\leq -\mu\|\nabla f(\boldsymbol{w}_t)\|_2^2 + \frac{L}{2}\mu^2(1+\gamma)(\sigma^2 + \|\nabla f(\boldsymbol{w}_t)\|_2^2) \\ &= -(\mu - \frac{L}{2}\mu^2(1+\gamma))\|\nabla f(\boldsymbol{w}_t)\|_2^2 + \frac{L}{2}\mu^2(1+\gamma)\sigma^2.\end{aligned}$$

Following same argument as before,

$$\begin{aligned}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\big[\|\nabla f(\boldsymbol{w}_t)\|_2^2\big] &\leq \frac{2(f(\boldsymbol{w}_0) - f^*)}{T(2\mu - L\mu^2(1+\gamma))} \\ &\quad + \frac{L\mu^2(1+\gamma)}{2\mu - L\mu^2(\gamma+1)}\sigma^2.\end{aligned}$$

It can be verified that if $T > 4L^2(\gamma+1)^2$, we can find $\mu \leq 2/\sqrt{T}$ such that $2\mu - L\mu^2(\gamma+1) =$

[2]In this case, it is not necessary to assume that the cost function has bounded gradient everywhere.

$2/\sqrt{T}$. This simplifies the above equation to

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\big[\|\nabla f(\boldsymbol{w}_t)\|_2^2\big] \leq \frac{f(\boldsymbol{w}_0)-f^*}{\sqrt{T}} + \frac{2L(1+\gamma)}{\sqrt{T}}\sigma^2.$$

The analysis for MMSE-QCS is straightforward. Note that $\widehat{\boldsymbol{g}}_{mmse} = \frac{1}{\gamma+1}\widehat{\boldsymbol{g}}_u$, where $\widehat{\boldsymbol{g}}_{mmse}$ is the MMSE-QCS quantized SG and $\widehat{\boldsymbol{g}}_u$ is the output of Unbiased-QCS. Therefore, training with MMSE-QCS and step-size μ would be the same as using Unbiased-QCS with step-size $\mu/(\gamma+1)$.

Remark 13. Note that since Unbiased-QCS has bounded variance and is unbiased, the compressed SG will be stochastic gradient itself with bounded variance. Hence, majority of the results can be readily applied to prove the convergence of Unbiased-QCS and MMSE-QCS under different conditions such as [98, 119].

D.10 Proof of Lemma 11

The proof is based on the ideas from [28] and follows the similar arguments with slight modifications, which is repeated here for the sake of completeness.

Let $\widetilde{\boldsymbol{w}}_t = \boldsymbol{w}_t - \mu\boldsymbol{r}_t$. Note that since by Lemma 9 the residue signal has bounded variance, $\widetilde{\boldsymbol{w}}_t$ would be bounded. It can be easily verified that $\widetilde{\boldsymbol{w}}_{t+1} = \widetilde{\boldsymbol{w}}_t - \mu\boldsymbol{g}_t$. Hence, following similar argument as in [28], for arbitrary $\rho > 0$

$$\begin{aligned}
&\mathbb{E}[f(\widetilde{\boldsymbol{w}}_{t+1})] - f(\widetilde{\boldsymbol{w}}_t)\\
&\leq \frac{L}{2}\,\mathbb{E}\big[\|\widetilde{\boldsymbol{w}}_{t+1}-\widetilde{\boldsymbol{w}}_t\|_2^2\big] + \langle\nabla f(\widetilde{\boldsymbol{w}}_t), \mathbb{E}[\widetilde{\boldsymbol{w}}_{t+1}-\widetilde{\boldsymbol{w}}_t]\rangle\\
&\leq \frac{L}{2}\mu^2 B - \mu(1-\rho)\|\nabla f(\boldsymbol{w}_t)\|_2^2 + \frac{1}{\rho}L^2\mu^3\,\mathbb{E}\big[\|\boldsymbol{r}_t\|_2^2\big].
\end{aligned}$$

On the other hand, from Lemma 9, the residue is bounded as $\mathbb{E}[\|\boldsymbol{r}_t\|_2^2] \leq \eta B$ where η is a constant depending on the β (weight of error feedback) and γ, according to (4.15) or (4.16)

of the main paper for Unbiased-QCS and MMSE-QCS. Therefore,

$$\mu(1-\rho)\|\nabla f(\boldsymbol{w}_t)\|_2^2 \leq \big(\frac{L}{2}\mu^2 + \frac{1}{\rho}L^2\mu^3\eta\big)B + f(\widetilde{\boldsymbol{w}}_t) - \mathbb{E}[f(\widetilde{\boldsymbol{w}}_{t+1})].$$

Taking expectation, rearranging terms and noting that $\widetilde{\boldsymbol{w}}_0 = \boldsymbol{w}_0$ and $\mathbb{E}[f(\boldsymbol{w}_T)] \geq f^*$, we conclude that for $0 < \rho < 1$,

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\big[\|\nabla f(\boldsymbol{w}_t)\|_2^2\big] \leq \frac{f(\boldsymbol{w}_0) - f^*}{T\mu(1-\rho)} + \frac{LB}{1-\rho}\big(\frac{\mu}{2} + \frac{L\eta\mu^2}{\rho}\big).$$

Setting $\rho = 0.5$, gives the desired result.

For tighter analysis, let μ and ρ be such that $\mu(1-\rho) = 1/\sqrt{T}$ and $\frac{\mu}{1-\rho} = \frac{1+\epsilon}{\sqrt{T}}$ for arbitrary $\epsilon > 0$, i.e.,

$$\mu = \frac{\sqrt{1+\epsilon}}{\sqrt{T}}, \quad 1-\rho = \frac{1}{\sqrt{1+\epsilon}}.$$

Therefore,

$$\begin{aligned}\min_t \mathbb{E}\big[\|\nabla f(\boldsymbol{w}_t)\|_2^2\big] \leq& \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\big[\|\nabla f(\boldsymbol{w}_t)\|_2^2\big] \\ \leq& \frac{f(\boldsymbol{w}_0) - f^* + \frac{L}{2}B}{\sqrt{T}} + L^2B\frac{(1+\epsilon)^2}{\sqrt{1+\epsilon}-1}\frac{\eta}{T}.\end{aligned}$$

D.11 Proof of Lemma 15

The solution of

$$\min_{\boldsymbol{x}} \|\boldsymbol{y} - \boldsymbol{x}\|_2^2 + \eta_1\|\boldsymbol{x}\|_0 + \eta_2\|\boldsymbol{x}_{\setminus j}\|_0, \tag{D.17}$$

is given by element-wise hard-thresholding $\boldsymbol{y}$, i.e.,

$$x_n = \begin{cases} 0 & \text{if } |y_n| \leq \sqrt{\eta} \\ y_n & \text{o.w.} \end{cases} \tag{D.18}$$

where $\eta = \eta_1$ or $\eta_1 + \eta_2$, depending on whether neuron n is in $\boldsymbol{y}_{\setminus j}$ or not.

Proof. Let Ω be the indexes in the j-th block. Therefore, $\boldsymbol{x}_{\setminus j}$ consists of elements of $\boldsymbol{x}$ that are not in the set Ω, and

$$\|\boldsymbol{y}-\boldsymbol{x}\|_2^2+\eta_1\|\boldsymbol{x}\|_0+\eta_2\|\boldsymbol{x}_{\setminus j}\|_0=\sum_{n\in\Omega}(y_n-x_n)^2+\eta_1\mathbb{I}(x_n\neq 0)$$
$$+\sum_{n\notin\Omega}(y_n-x_n)^2+(\eta_1+\eta_2)\mathbb{I}(x_n\neq 0),$$

where $\mathbb{I}(z)=1$ if z is true and is 0 otherwise. Therefore, the minimization in (D.17) can be cast as separate minimizations over scalars x_n. For example, if $n\in\Omega$, there are two possibilities for x_n,

$$\begin{cases} x_n=0 & \Rightarrow cost=y_n^2 \\ x_n\neq 0 & \Rightarrow cost=\min_{x_n\neq 0}(y_n-x_n)^2+\eta_1=\eta_1 \end{cases}$$

Hence, the solution would be

$$n\in\Omega: \quad x_n^*=\begin{cases} 0 & \text{if } |y_n|\leq\sqrt{\eta_1} \\ y_n & \text{o.w.} \end{cases}$$

Similarly,

$$n\notin\Omega: \quad x_n^*=\begin{cases} 0 & \text{if } |y_n|\leq\sqrt{\eta_1+\eta_2} \\ y_n & \text{o.w.} \end{cases}$$

■

D.12 Proof of Theorem 16

First, we note that for any permutation matrix $\boldsymbol{\Pi}$, $\|\widehat{\boldsymbol{W}}-\boldsymbol{W}\boldsymbol{\Pi}^{\mathsf{T}}\|_F^2=\|\widehat{\boldsymbol{W}}\boldsymbol{\Pi}-\boldsymbol{W}\|_F^2$, $\|\widehat{\boldsymbol{W}}\|_0=\|\widehat{\boldsymbol{W}}\boldsymbol{\Pi}\|_0$, and $\|\boldsymbol{M}\odot\widehat{\boldsymbol{W}}\|_0=\|(\boldsymbol{M}\boldsymbol{\Pi})\odot(\widehat{\boldsymbol{W}}\boldsymbol{\Pi})\|_0$. Therefore, by defining

$\boldsymbol{X} = \widehat{\boldsymbol{W}}\boldsymbol{\Pi}$, the optimization (6.6) can be rewritten as

$$\begin{aligned}
&\min_{\boldsymbol{\Pi}} \min_{\boldsymbol{X}} \|\boldsymbol{X} - \boldsymbol{W}\|_F^2 + \eta_1 \|\boldsymbol{X}\|_0 + \eta_2 \|(\boldsymbol{M}\boldsymbol{\Pi}) \odot \boldsymbol{X}\|_0 \\
=&\min_{\boldsymbol{\Pi}} \min_{\boldsymbol{X}} \sum_{i,k:\, \Pi_{k,i}=1} \|\boldsymbol{x}_i - \boldsymbol{w}_i\|_2^2 + \eta_1 \|\boldsymbol{x}_i\|_0 + \eta_2 \|\boldsymbol{m}_k \odot \boldsymbol{x}_i\|_0, \\
=&\min_{\boldsymbol{\Pi}} \sum_{i,k:\, \Pi_{k,i}=1} \min_{\boldsymbol{x}_i} \|\boldsymbol{x}_i - \boldsymbol{w}_i\|_2^2 + \eta_1 \|\boldsymbol{x}_i\|_0 + \eta_2 \|\boldsymbol{m}_k \odot \boldsymbol{x}_i\|_0.
\end{aligned}$$

On the other hand, recall that $\boldsymbol{M} = 1 - \operatorname{diag}\big(\mathbf{1}_{\iota_1 \times n_1}, \ldots, \mathbf{1}_{\iota_P \times n_P}\big)$, and hence if $\boldsymbol{m}_k$ is from the j-th sub-block, i.e., it corresponds to the j-th worker, the inner minimization would be

$$C_{ji} = \min_{\boldsymbol{x}_i} \|\boldsymbol{x}_i - \boldsymbol{w}_i\|_2^2 + \eta_1 \|\boldsymbol{x}_i\|_0 + \eta_2 \|\boldsymbol{x}_{i,\setminus j}\|_0. \tag{D.19}$$

By repeating the k-th row of matrix $\boldsymbol{C}$ whose elements are defined as (D.19) to construct the new $N \times N$ matrix $\widetilde{\boldsymbol{C}}$, we will have $C_{ji} = \tilde{C}_{ki}$. Therefore,

$$\min_{\widehat{\boldsymbol{W}}, \boldsymbol{\Pi}} \|\widehat{\boldsymbol{W}} - \boldsymbol{W}\boldsymbol{\Pi}^{\mathsf{T}}\|_F^2 + \eta_1 \|\widehat{\boldsymbol{W}}\|_0 + \eta_2 \|\boldsymbol{M} \odot \widehat{\boldsymbol{W}}\|_0 = \min_{\boldsymbol{\Pi}} \sum_{(i,k):\, \Pi_{k,i}=1} \tilde{C}_{ki}.$$

As a result, selecting the best neuron assignment boils down to choosing N elements from $\widetilde{\boldsymbol{C}}$ such that from each row or column only one element is selected and the sum of the selected values is minimum. This problem can be solved efficiently in polynomial time using the Hungarian algorithm. [120, 121] solve the assignment algorithm with $\mathcal{O}(N^3)$ time complexity. Since the complexity of creating $\widetilde{\boldsymbol{C}}$ is at most $\mathcal{O}(N^2)$, the total complexity of Algorithm 1 would be $\mathcal{O}(N^3)$.

D.13 Proof of Theorem 17

Let $\boldsymbol{x}^{(l)}$ and $\widehat{\boldsymbol{x}}^{(l)}$ be the signals in the original and modified neural network, corresponding to the input $\boldsymbol{x}$. Note that $\boldsymbol{\Pi}^{(0)} = \boldsymbol{I}$ and the input to both networks are the same, $\boldsymbol{x}^{(1)} = \widehat{\boldsymbol{x}}^{(1)} = \boldsymbol{x}$. Let $\boldsymbol{\Pi}^{(l)}$ and $\{\widehat{\boldsymbol{W}}^{(l)}, \widehat{\boldsymbol{b}}^{(l)}\}$ be the permutation matrix and parameters of the modified neural

network, found via (6.3). Therefore, using $\boldsymbol{x}^{(l+1)} = \sigma((\boldsymbol{W}^{(l)})^\mathsf{T}\boldsymbol{x}^{(l)} + \boldsymbol{b}^{(l)})$, for any arbitrary layer l,

$$
\begin{aligned}
& \|\boldsymbol{\Pi}^{(l)}\boldsymbol{x}^{(l+1)} - \widehat{\boldsymbol{x}}^{(l+1)}\|_2 \\
= \; & \|\boldsymbol{\Pi}^{(l)}\sigma((\boldsymbol{W}^{(l)})^\mathsf{T}\boldsymbol{x}^{(l)} + \boldsymbol{b}^{(l)}) - \sigma((\widehat{\boldsymbol{W}}^{(l)})^\mathsf{T}\widehat{\boldsymbol{x}}^{(l)} + \widehat{\boldsymbol{b}}^{(l)})\|_2 \\
\overset{(a)}{=} \; & \|\sigma(\boldsymbol{\Pi}^{(l)}(\boldsymbol{W}^{(l)})^\mathsf{T}\boldsymbol{x}^{(l)} + \boldsymbol{\Pi}^{(l)}\boldsymbol{b}^{(l)}) - \sigma((\widehat{\boldsymbol{W}}^{(l)})^\mathsf{T}\widehat{\boldsymbol{x}}^{(l)} + \widehat{\boldsymbol{b}}^{(l)})\|_2 \\
\overset{(b)}{\leq} \; & \|(\boldsymbol{\Pi}^{(l)}(\boldsymbol{W}^{(l)})^\mathsf{T}\boldsymbol{x}^{(l)} + \boldsymbol{\Pi}^{(l)}\boldsymbol{b}^{(l)}) - ((\widehat{\boldsymbol{W}}^{(l)})^\mathsf{T}\widehat{\boldsymbol{x}}^{(l)} + \widehat{\boldsymbol{b}}^{(l)})\|_2 \\
\overset{(c)}{=} \; & \|\boldsymbol{\Pi}^{(l)}(\boldsymbol{W}^{(l)})^\mathsf{T}\boldsymbol{x}^{(l)} - (\widehat{\boldsymbol{W}}^{(l)})^\mathsf{T}\widehat{\boldsymbol{x}}^{(l)}\|_2 \\
= \; & \|\big(\boldsymbol{\Pi}^{(l-1)}\boldsymbol{W}^{(l)}(\boldsymbol{\Pi}^{(l)})^\mathsf{T} - \widehat{\boldsymbol{W}}^{(l)}\big)^\mathsf{T}\widehat{\boldsymbol{x}}^{(l)} + \big(\boldsymbol{W}^{(l)}(\boldsymbol{\Pi}^{(l)})^\mathsf{T}\big)^\mathsf{T}\big((\boldsymbol{\Pi}^{(l-1)})^\mathsf{T}\widehat{\boldsymbol{x}}^{(l)} - \boldsymbol{x}^{(l)}\big)\|_2 \\
\leq \; & \|\big(\boldsymbol{\Pi}^{(l-1)}\boldsymbol{W}^{(l)}(\boldsymbol{\Pi}^{(l)})^\mathsf{T} - \widehat{\boldsymbol{W}}^{(l)}\big)^\mathsf{T}\widehat{\boldsymbol{x}}^{(l)}\|_2 + \|\big(\boldsymbol{W}^{(l)}(\boldsymbol{\Pi}^{(l)})^\mathsf{T}\big)^\mathsf{T}\big((\boldsymbol{\Pi}^{(l-1)})^\mathsf{T}\widehat{\boldsymbol{x}}^{(l)} - \boldsymbol{x}^{(l)}\big)\|_2 \\
\overset{(d)}{\leq} \; & \|\boldsymbol{\Pi}^{(l-1)}\boldsymbol{W}^{(l)}(\boldsymbol{\Pi}^{(l)})^\mathsf{T} - \widehat{\boldsymbol{W}}^{(l)}\|_F\,\|\widehat{\boldsymbol{x}}^{(l)}\|_2 + \|\boldsymbol{W}^{(l)}(\boldsymbol{\Pi}^{(l)})^\mathsf{T}\|_F\,\|(\boldsymbol{\Pi}^{(l-1)})^\mathsf{T}\widehat{\boldsymbol{x}}^{(l)} - \boldsymbol{x}^{(l)}\|_2 \\
= \; & \|\widehat{\boldsymbol{W}}^{(l)} - \boldsymbol{\Pi}^{(l-1)}\boldsymbol{W}^{(l)}(\boldsymbol{\Pi}^{(l)})^\mathsf{T}\|_F\,\|\widehat{\boldsymbol{x}}^{(l)}\|_2 + \|\boldsymbol{W}^{(l)}\|_F\,\|\widehat{\boldsymbol{x}}^{(l)} - \boldsymbol{\Pi}^{(l-1)}\boldsymbol{x}^{(l)}\|_2 \\
\overset{(e)}{\leq} \; & \varepsilon\|\widehat{\boldsymbol{x}}^{(l)}\|_2 + \tau\|\boldsymbol{\Pi}^{(l-1)}\boldsymbol{x}^{(l)} - \widehat{\boldsymbol{x}}^{(l)}\|_2 \\
\leq \; & \varepsilon\big(\|\widehat{\boldsymbol{x}}^{(l)} - \boldsymbol{\Pi}^{(l-1)}\boldsymbol{x}^{(l)}\|_2 + \|\boldsymbol{\Pi}^{(l-1)}\boldsymbol{x}^{(l)}\|_2\big) + \tau\|\boldsymbol{\Pi}^{(l-1)}\boldsymbol{x}^{(l)} - \widehat{\boldsymbol{x}}^{(l)}\|_2 \\
= \; & (\tau+\varepsilon)\|\boldsymbol{\Pi}^{(l-1)}\boldsymbol{x}^{(l)} - \widehat{\boldsymbol{x}}^{(l)}\|_2 + \varepsilon\|\boldsymbol{x}^{(l)}\|_2 \\
\leq \; & (\tau+\varepsilon)\|\boldsymbol{\Pi}^{(l-1)}\boldsymbol{x}^{(l)} - \widehat{\boldsymbol{x}}^{(l)}\|_2 + \varepsilon B
\end{aligned}
$$

where (a) is because $\boldsymbol{\Pi}\sigma(\boldsymbol{z}) = \sigma(\boldsymbol{\Pi}\boldsymbol{z})$ for arbitrary permutation $\boldsymbol{\Pi}$ and vector $\boldsymbol{z}$, (b) is because $\sigma(\cdot)$ is 1-Lipschitz, (c) is due to the fact that $\widehat{\boldsymbol{b}}^{(l)} = \boldsymbol{\Pi}^{(l)}\boldsymbol{b}^{(l)}$, (d) is from $\|\boldsymbol{A}\boldsymbol{z}\|_2 \leq \|\boldsymbol{A}\|_2\|\boldsymbol{z}\|_2 \leq \|\boldsymbol{A}\|_F\|\boldsymbol{z}\|_2$ for arbitrary $\boldsymbol{A}$ and $\boldsymbol{z}$, and (e) is by assumption **A2** and (6.3). Therefore,

$$
\|\boldsymbol{\Pi}^{(l)}\boldsymbol{x}^{(l+1)} - \widehat{\boldsymbol{x}}^{(l+1)}\|_2 \leq (\tau+\epsilon)\|\boldsymbol{\Pi}^{(l-1)}\boldsymbol{x}^{(l)} - \widehat{\boldsymbol{x}}^{(l)}\|_2 + \varepsilon B. \tag{D.20}
$$

Since $\boldsymbol{x} = \boldsymbol{\Pi}^{(0)}\boldsymbol{x}^{(1)} = \widehat{\boldsymbol{x}}^{(1)}$, (D.20) implies that

$$\|\boldsymbol{\Pi}^{(l)}\boldsymbol{x}^{(l+1)} - \widehat{\boldsymbol{x}}^{(l+1)}\|_2 \leq \big(\sum_{k=1}^{l}(\tau+\epsilon)^{l-k}\big)\varepsilon B = \frac{(\tau+\varepsilon)^l - 1}{\tau+\varepsilon-1}\varepsilon B. \tag{D.21}$$

Specifically, for the output signals, $\boldsymbol{y} = \boldsymbol{x}^{L+1}$ and $\widehat{\boldsymbol{y}} = \widehat{\boldsymbol{x}}^{(L+1)}$, it implies that

$$\|\widehat{\boldsymbol{y}} - \boldsymbol{\Pi}\boldsymbol{y}\|_2 \leq \varepsilon\frac{(\tau+\varepsilon)^L - 1}{\tau+\varepsilon-1}B.$$

REFERENCES

[1] M. A. Zinkevich, A. J. Smola, M. Weimer, L. Li, and A. J. Smola, "Parallelized Stochastic Gradient Descent," in *Advances in Neural Information Processing Systems 23*, J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, Eds., Curran Associates, Inc., 2010, pp. 2595–2603.

[2] M. F. Balcan, A. Blum, S. Fine, and Y. Mansour, "Distributed learning, communication complexity and privacy," in *Conference on Learning Theory*, 2012, pp. 26–1.

[3] T. Chilimbi, Y. Suzue, J. Apacible, and K. Kalyanaraman, "Project Adam: Building an Efficient and Scalable Deep Learning Training System," in *Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'14, 2014, pp. 571–582, ISBN: 9781931971164.

[4] P. Watcharapichat, V. L. Morales, R. C. Fernandez, and P. Pietzuch, "Ako: Decentralised Deep Learning with Partial Gradient Exchange," *SoCC '16*, no. i, pp. 84–97, 2016.

[5] V. Smith, S. Forte, C. Ma, M. Takac, M. I. Jordan, M. Jaggi, M. Chenxin, M. Takáč, M. I. Jordan, and M. Jaggi, "CoCoA: A General Framework for Communication-Efficient Distributed Optimization," *Journal of Machine Learning Research*, vol. 18, pp. 1–49, 2016. arXiv: 1611.02189.

[6] W. Wang, M. Zhang, G. Chen, H. V. Jagadish, B. C. Ooi, and K.-L. Tan, "Database Meets Deep Learning: Challenges and Opportunities," *SIGMOD Rec.*, vol. 45, no. 2, pp. 17–22, Sep. 2016.

[7] W. Wang, G. Chen, H. Chen, T. T. A. Dinh, J. Gao, B. C. Ooi, K.-L. Tan, S. Wang, and M. Zhang, "Deep Learning at Scale and at Ease," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 12, no. 4s, 69:1–69:25, Nov. 2016.

[8] S. Gupta, W. Zhang, and F. Wang, "Model accuracy and runtime tradeoff in distributed deep learning: A systematic study," in *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, IEEE, 2016, pp. 171–180, ISBN: 9780999241103. arXiv: arXiv:1509.04210v3.

[9] F. Seide, H. Fu, J. Droppo, G. Li, D. Yu, M. Stevenson, R. Winter, and B. Widrow, "1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs," in *Interspeech*, 2014, pp. 1058–1062, ISBN: 9781510810587.

[10] N. Dryden, S. A. Jacobs, T. Moon, and B. Van Essen, "Communication quantization for data-parallel training of deep neural networks," in *Proceedings of the Workshop*

on Machine Learning in High Performance Computing Environments, ser. MLHPC '16, Salt Lake City, Utah: IEEE Press, 2016, pp. 1–8, ISBN: 978-1-5090-3882-4.

[11] C.-Y. Chen, J. Choi, D. Brand, A. Agrawal, W. Zhang, and K. Gopalakrishnan, "AdaComp : Adaptive Residual Gradient Compression for Data-Parallel Distributed Training," *Thirty-Second AAAI Conference on Artificial Intelligence*, pp. 2827–2835, 2017. arXiv: 1712.02679.

[12] T. T. Doan, S. T. Maguluri, and J. Romberg, "Fast convergence rates of distributed subgradient methods with adaptive quantization," *arXiv preprint arXiv:1810.13245*, 2018.

[13] 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*, ser. Proceedings of Machine Learning Research, PMLR, 2018, pp. 560–569.

[14] M. Safaryan and P. Richtárik, "On stochastic sign descent methods," *arXiv preprint arXiv:1905.12938*, 2019.

[15] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-Efficient SGD via Gradient Quantization and Encoding," in *Advances in Neural Information Processing Systems*, 2017, pp. 1707–1718. arXiv: 1610.02132.

[16] W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, and H. Li, "TernGrad: Ternary Gradients to Reduce Communication in Distributed Deep Learning," in *Advances in neural information processing systems*, Curran Associates, Inc., 2017, pp. 1509–1519. arXiv: 1705.07878.

[17] N. Strom, "Scalable distributed DNN training using commodity GPU cloud computing.," in *INTERSPEECH*, vol. 7, 2015, p. 10.

[18] A. F. Aji and K. Heafield, "Sparse communication for distributed gradient descent," *arXiv preprint arXiv:1704.05021*, 2017. arXiv: arXiv:1704.05021v2.

[19] S. U. Stich, J.-B. Cordonnier, and M. Jaggi, "Sparsified sgd with memory," in *Advances in Neural Information Processing Systems*, 2018, pp. 4452–4463. arXiv: 1809.07599.

[20] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Sparse binary compression: Towards distributed deep learning with minimal communication," *arXiv preprint, arXiv 1805.08768*, vol. abs/1805.08768, 2018. arXiv: 1805.08768.

[21] D. Alistarh, T. Hoefler, M. Johansson, N. Konstantinov, S. Khirirat, and C. Renggli, "The convergence of sparsified gradient methods," in *Advances in Neural Information Processing Systems*, 2018, pp. 5977–5987.

[22] S. Shi, Q. Wang, K. Zhao, Z. Tang, Y. Wang, X. Huang, and X. Chu, "A distributed synchronous sgd algorithm with global top-k sparsification for low bandwidth networks," *arXiv preprint arXiv:1901.04359*, 2019.

[23] Y. Lin, S. Han, H. Mao, Y. Wang, B. Dally, and W. J. Dally, "Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training," in *International Conference on Learning Representations*, 2018, pp. 1–13.

[24] C. Renggli, D. Alistarh, T. Hoefler, and M. Aghagolzadeh, "SparCML: High-performance sparse communication for machine learning," *arXiv preprint arXiv:1802.08021*, 2018.

[25] J. Wangni, J. Wang, J. Liu, and T. Zhang, "Gradient sparsification for communication-efficient distributed optimization," in *Advances in Neural Information Processing Systems*, 2018, pp. 1306–1316. arXiv: 1710.09854.

[26] H. Wang, S. Sievert, Z. Charles, S. Liu, S. Wright, and D. Papailiopoulos, "ATOMO: Communication-efficient Learning via Atomic Sparsification," in *Advances in Neural Information Processing Systems 31*, Curran Associates, Inc., 2018, pp. 9850–9861. arXiv: 1806.04090.

[27] J. Wu, W. Huang, J. Huang, and T. Zhang, "Error Compensated Quantized SGD and its Applications to Large-scale Distributed Optimization," *ICML*, 2018. arXiv: 1806.08054.

[28] S. P. Karimireddy, Q. Rebjock, S. U. Stich, and M. Jaggi, "Error Feedback Fixes SignSGD and other Gradient Compression Schemes," *arXiv preprint arXiv:1901.09847*, 2019.

[29] A. Øland and B. Raj, "Reducing communication overhead in distributed learning by an order of magnitude (almost)," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Apr. 2015, pp. 2219–2223, ISBN: 9781467369978.

[30] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Transactions on Automatic Control*, vol. 31, no. 9, pp. 803–812, Sep. 1986.

[31] F. Niu, B. Recht, C. Ré, and S. Wright, "Hogwild: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent," in *Advances in Neural Information Processing Systems 24*, 2011, pp. 693–701.

[32] T. Paine, H. Jin, J. Yang, Z. Lin, and T. Huang, "GPU Asynchronous Stochastic Gradient Descent to Speed Up Neural Network Training," *arXiv preprint, arXiv:1312.6186*, pp. 1–6, 2013. arXiv: 1312.6186.

[33] M. Wang, H. Zhou, M. Guo, and Z. Zhang, "A scalable and topology configurable protocol for distributed parameter synchronization," in *Proceedings of 5th Asia-Pacific Workshop on Systems*, ACM, 2014, p. 13.

[34] L. Wang, Y. Yang, M. R. Min, and S. Chakradhar, "Accelerating Deep Neural Network Training with Inconsistent Stochastic Gradient Descent," *arXiv preprint arXiv:1603.05544*, 2016.

[35] S.-Y. Zhao and W.-J. Li, "Fast Asynchronous Parallel Stochastic Gradient Decent," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016, pp. 1–15. arXiv: 1508.05711.

[36] A. Odena, "Faster Asynchronous SGD," *arXiv preprint arXiv:1601.04033*, 2016.

[37] S. Dutta, G. Joshi, S. Ghosh, P. Dube, and P. Nagpurkar, "Slow and Stale Gradients Can Win the Race: Error-Runtime Trade-offs in Distributed SGD," *arXiv preprint arXiv:1803.01113*, 2018.

[38] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le, *et al.*, "Large scale distributed deep networks," in *Advances in neural information processing systems*, 2012, pp. 1223–1231.

[39] Q. Ho, J. Cipar, H. Cui, S. Lee, J. K. Kim, P. B. Gibbons, G. A. Gibson, G. Ganger, and E. Xing, "More Effective Distributed ML via a Stale Synchronous Parallel Parameter Server," in *Advances in Neural Information Processing Systems 26*, 2013, pp. 1223–1231.

[40] W. Zhang, S. Gupta, X. Lian, and J. Liu, "Staleness-Aware Async-SGD for Distributed Deep Learning," in *International Joint Conference on Artificial Intelligence, IJCAI*, 2016, pp. 2350–2356.

[41] N. Ferdinand and S. C. Draper, "Anytime Stochastic Gradient Descent: A Time to Hear from all the Workers," in *2018 56th Annual Allerton Conference on Communication, Control, and Computing, Allerton 2018*, vol. 2, 2019, pp. 552–559, ISBN: 9781538665961. arXiv: arXiv:1810.02976v1.

[42] H. Zhang, C. J. Hsieh, and V. Akella, "HogWild++: A New Mechanism for Decentralized Asynchronous Stochastic Gradient Descent," in *2016 IEEE 16th International Conference on Data Mining (ICDM)*, Dec. 2016, pp. 629–638.

[43] S. U. Stich, "Local SGD Converges Fast and Communicates Little," in *ICLR*, 2019, pp. 1–12. arXiv: 1805.09767.

[44] F. Zhou and G. Cong, "On the convergence properties of a k-step averaging stochastic gradient descent algorithm for nonconvex optimization," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, International Joint Conferences on Artificial Intelligence Organization, Jul. 2018.

[45] H. Yu, S. Yang, and S. Zhu, "Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 5693–5700.

[46] D. Slepian and J. Wolf, "Noiseless coding of correlated information sources," *IEEE Transactions on Information Theory*, vol. 19, no. 4, pp. 471–480, Jul. 1973.

[47] T. Berger, Z. Zhang, and H. Viswanathan, "The CEO problem," *IEEE Transactions on Information Theory*, vol. 42, no. 3, pp. 887–902, May 1996.

[48] H. Viswanathan and T. Berger, "The quadratic Gaussian CEO problem," *IEEE Transactions on Information Theory*, vol. 43, no. 5, pp. 1549–1559, 1997.

[49] J. Chen and T. Berger, "Successive Wyner-Ziv Coding Scheme and Its Application to the Quadratic Gaussian CEO Problem," *IEEE Transactions on Information Theory*, vol. 54, no. 4, pp. 1586–1603, Apr. 2008.

[50] A. Abdi and F. Fekri, "Nested dithered quantization for communication reduction in distributed training," *arXiv preprint arXiv:1904.01197*, 2019.

[51] ——, "Reducing communication overhead via CEO in distributed training," in *IEEE International Workshop on Signal Processing Advances in Wireless Communications*, 2019, pp. 1–5.

[52] A. Abdi and F. Fekri, "Indirect stochastic gradient quantization and its application in distributed deep learning," in *AAAI conference on Artificial Intelligence*, 2020.

[53] A. Abdi and F. Fekri, "Quantized compressive sampling of stochastic gradients for efficient communication in distributed deep learning," in *AAAI conference on Artificial Intelligence*, 2020.

[54] J. Konecný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," in *NIPS Workshop on Private Multi-Party Machine Learning*, 2016.

[55] J. Konecny, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv preprint arXiv:1610.02527*, 2016.

[56] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," *arXiv preprint, arXiv:1602.05629v1*, 2016.

[57] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, *et al.*, "Communication-efficient learning of deep networks from decentralized data," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.

[58] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[59] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 2, pp. 1–19, Jan. 2019.

[60] M. Goldenbaum and S. Stanczak, "Robust analog function computation via wireless multiple-access channels," *IEEE Transactions on Communications*, vol. 61, no. 9, pp. 3863–3877, Sep. 2013.

[61] M. Goldenbaum, H. Boche, and S. Stańczak, "Harnessing interference for analog function computation in wireless sensor networks," *IEEE Transactions on Signal Processing*, vol. 61, no. 20, pp. 4893–4906, 2013.

[62] A. Abdi, Y. M. Saidutta, and F. Fekri, "Analog compression and communication for federated learning over wireless MAC," in *IEEE International Workshop on Signal Processing Advances in Wireless Communications*, 2020.

[63] I.-H. H. Chung, T. N. Sainath, B. Ramabhadran, M. Picheny, J. Gunnels, V. Austel, U. Chauhari, and B. Kingsbury, "Parallel Deep Neural Network Training for Big Data on Blue Gene/Q," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '14, vol. 28, Piscataway, NJ, USA: IEEE Press, 2014, pp. 745–753, ISBN: 978-1-4799-5500-8.

[64] M. D. F. De Grazia, I. Stoianov, and M. Zorzi, "Parallelization of deep networks," *Proceedings of 2012 European Symposium on Artificial NN, Computational Intelligence and Machine Learning*, no. April, pp. 621–626, 2012.

[65] G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," *ArXiv e-prints*, pp. 1–9, Mar. 2015. arXiv: 1503.02531.

[66] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," in *ICLR*, 2015.

[67] S. Zagoruyko and N. Komodakis, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," in *ICLR*, 2017.

[68] Y. Cheng, X. Y. Felix, R. S. Feris, S. Kumar, A. Choudhary, and S.-F. Chang, "Fast neural networks with circulant projections," *arXiv preprint arXiv:1502.03436*, 2015.

[69] Z. Yang, M. Moczulski, M. Denil, N. D. Freitas, A. Smola, L. Song, Z. Wang, N. de Freitas, A. Smola, L. Song, and Z. Wang, "Deep Fried Convnets," in *The IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015, pp. 1476–1483, ISBN: 9781467383912. arXiv: arXiv:1412.7149v4.

[70] R. Rigamonti, A. Sironi, V. Lepetit, and P. Fua, "Learning separable filters," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, Jun. 2013.

[71] C. Tai, T. Xiao, Y. Zhang, X. Wang, *et al.*, "Convolutional neural networks with low-rank regularization," in *ICLR*, 2016.

[72] H. Zhou, J. M. Alvarez, and F. Porikli, "Less is more: Towards compact cnns," in *European Conference on Computer Vision*, Springer, 2016, pp. 662–677.

[73] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 2074–2082, ISBN: 1878-3686 (Electronic). arXiv: 1608.03665.

[74] C. Louizos, M. Welling, and D. P. Kingma, "Learning Sparse Neural Networks through ℓ_0 Regularization," in *ICLR*, 2018, pp. 1–13.

[75] Y. L. Cun, J. S. Denker, S. A. Sola, T. B. Laboratories, and S. A. Solla, "Optimal Brain Damage," in *Advances in Neural Information Processing Systems 2*, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1990, pp. 598–605, ISBN: 1-55860-100-7.

[76] B. Hassibi, D. G. Stork, S. H. Road, and M. Park, "Second Order Derivatives for Network Pruning: Optimal Brain Surgeon," in *Advances in Neural Information Processing Systems 5, [NIPS Conference]*, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993, pp. 164–171, ISBN: 1-55860-274-7.

[77] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both Weights and Connections for Efficient Neural Networks," *CoRR*, vol. abs/1506.02626, pp. 1–9, 2015. arXiv: 1506.02626.

[78] G. Castellano, A. M. Fanelli, and M. Pelillo, "An iterative pruning algorithm for feedforward neural networks," *IEEE Transactions on Neural Networks*, vol. 8, no. 3, pp. 519–531, May 1997.

[79] C.-S. S. Leung, K.-W. W. Wong, P.-F. F. Sum, and L.-W. W. Chan, "A pruning method for the recursive least squared algorithm," *Neural Networks*, vol. 14, no. 2, pp. 147–174, 2001.

[80] A. Aghasi, A. Abdi, N. Nguyen, and J. Romberg, "Net-Trim: Convex Pruning of Deep Neural Networks with Performance Guarantee," in *Advances in Neural Information Processing Systems 30*, Nips, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., Curran Associates, Inc., 2017, pp. 3180–3189.

[81] A. Aghasi, A. Abdi, and J. Romberg, "Fast convex pruning of deep neural networks," *SIAM Journal on Mathematics of Data Science*, vol. 2, no. 1, pp. 158–188, 2020.

[82] Y. Gong, L. Liu, M. Yang, and L. D. Bourdev, "Compressing Deep Convolutional Networks using Vector Quantization," *CoRR*, vol. abs/1412.6115, pp. 1–10, 2014. arXiv: 1412.6115.

[83] S. Anwar, K. Hwang, and W. Sung, "Fixed point optimization of deep convolutional neural networks for object recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, Apr. 2015, pp. 1131–1135, ISBN: 9781467369978.

[84] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Advances in Neural Information Processing Systems*, 2015, pp. 3123–3131, ISBN: 1872-2075. arXiv: 1511.00363.

[85] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized Neural Networks," in *Advances in Neural Information Processing Systems 29*, Curran Associates, Inc., 2016, pp. 4107–4115. arXiv: 1602.02505.

[86] D. Wan, F. Shen, L. Liu, F. Zhu, J. Qin, L. Shao, and H. Tao Shen, "Tbn: Convolutional neural network with ternary inputs and binary weights," in *The European Conference on Computer Vision (ECCV)*, Sep. 2018.

[87] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," in *ICLR*, 2017, pp. 1–10.

[88] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European Conference on Computer Vision*, Springer, 2016, pp. 525–542.

[89] A. Abdi, S. Rashidi, F. Fekri, and T. Krishna, "Restructuring, pruning, and adjustment of deep models for parallel distributed inference," in *submitted to Advances in Neural Information Processing Systems*, 2020.

[90] V. Prabhakaran, D. Tse, and K. Ramachandran, "Rate region of the quadratic Gaussian CEO problem," in *International Symposium on Information Theory, 2004. ISIT 2004. Proceedings.*, Jun. 2004, pp. 119–.

[91] L. Schuchman, "Dither signals and their effect on quantization noise," *IEEE Transactions on Communication Technology*, vol. 12, no. 4, pp. 162–165, 1964.

[92] R. Zamir, S. Shamai, and U. Erez, "Nested linear/lattice codes for structured multiterminal binning," *IEEE Transactions on Information Theory*, vol. 48, no. 6, pp. 1250–1276, Jun. 2002.

[93] S. S. Pradhan, J. Kusuma, and K. Ramchandran, "Distributed compression in a dense microsensor network," *IEEE Signal Processing Magazine*, vol. 19, no. 2, pp. 51–60, Mar. 2002.

[94] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[95] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[96] J. Max, "Quantizing for minimum distortion," *IRE Transactions on Information Theory*, vol. 6, no. 1, pp. 7–12, Mar. 1960.

[97] S. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, Mar. 1982.

[98] L. Bottou, "Online Learning and Stochastic Approximations, Revised 2018," *On-Line Learning in Neural Networks*, vol. 17, no. 9, pp. 1–35, 1998.

[99] R. Rojas, "Neural Networks, Sringer-Verlag, Berlin," *Neural Networks*, vol. 7, 1996.

[100] I. Goodfellow, Y. Bengio, A. Courville, and N. Okazaki, *Deep learning*. MIT press, 2016.

[101] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[102] R. M. Gray and T. G. Stockham, "Dithered quantizers," *IEEE Transactions on Information Theory*, vol. 39, no. 3, pp. 805–812, 1993.

[103] A. Neelakantan, L. Vilnis, Q. V. Le, I. Sutskever, L. Kaiser, K. Kurach, and J. Martens, "Adding gradient noise improves learning for very deep networks," *arXiv preprint*, pp. 1–11, 2015. arXiv: 1511.06807.

[104] H. Noh, T. You, J. Mun, and B. Han, "Regularizing Deep Neural Networks by Noise: Its Interpretation and Optimization," in *Advances in Neural Information Processing Systems*, 2017, pp. 5109–5118. arXiv: 1710.05179.

[105] G. Zhu, Y. Wang, and K. Huang, "Broadband analog aggregation for low-latency federated edge learning," *arXiv preprint arXiv:1812.11494 v3*, 2018.

[106] T. Sery and K. Cohen, "On analog gradient descent learning over multiple access fading channels," *arXiv preprint arXiv:1908.07463*, 2019.

[107] W. Liu and X. Zang, "Over-the-air computation systems: Optimization, analysis and scaling laws," *arXiv preprint arXiv:1909.00329*, 2019.

[108] N. H. Tran, W. Bao, A. Zomaya, N. M. N.H., and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, IEEE, Apr. 2019.

[109] M. M. Amiri and D. Gunduz, "Over-the-air machine learning at the wireless edge," in *2019 IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, IEEE, Jul. 2019.

[110] ——, "Machine learning at the wireless edge: Distributed stochastic gradient descent over-the-air," *arXiv preprint arXiv:1901.00844*, 2019.

[111] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, Mar. 1955.

[112] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, Mar. 1957.

[113] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[114] S. Rashidi, S. Sridharan, S. Srinivasan, and T. Krishna, "ASTRA-SIM: Enabling SW/HW Co-Design Exploration for Distributed DL Training Platforms," in *IEEE International Symposium on Performance Analysis of Systems and Software, ISPASS 2020*.

[115] NVIDIA, *Nvidia collective communications library (nccl)*, 2018.

[116] A. Samajdar, J. M. Joseph, Y. Zhu, P. Whatmough, M. Mattina, and T. Krishna, "A systematic methodology for characterizing scalability of dnn accelerators using scale-sim," in *2020 IEEE International Symposium on Performance Analysis of Systems and Software*, 2020.

[117] R. M. Gray and D. L. Neuhoff, "Quantization," *IEEE Transactions on Information Theory*, vol. 44, no. 6, pp. 2325–2383, Oct. 1998.

[118] S. Arora, Y. Liang, and T. Ma, "Why are deep nets reversible: A simple theory, with implications for training," *arXiv preprint arXiv:1511.05653*, pp. 1–23, 2015. arXiv: 1511.05653.

[119] S. Bubeck, "Convex optimization: Algorithms and complexity," *Foundations and Trends®in Machine Learning*, vol. 8, no. 3-4, pp. 231–357, 2015. arXiv: 1405.4980.

[120] N. Tomizawa, "On some techniques useful for solution of transportation network problems," *Networks*, vol. 1, no. 2, pp. 173–194, 1971.

[121] R. Jonker and A. Volgenant, "A shortest augmenting path algorithm for dense and sparse linear assignment problems," *Computing*, vol. 38, no. 4, pp. 325–340, 1987.