

# 1 Quantized Federated Learning

---

Nir Shlezinger, Mingzhe Chen, Yonina C. Eldar, H. Vincent Poor, and Shuguang Cui

## 1.1 Introduction

Recent years have witnessed unprecedented success of machine learning methods in a broad range of applications [1]. These systems utilize highly parameterized models, such as deep neural networks, trained using a massive amount of labeled data samples. In many applications, samples are available at remote users, e.g., smartphones and other edge devices, and the common strategy is to gather these samples at a computationally powerful server, where the model is trained [2]. Often, data sets, such as images and text messages, contain private information, and thus the user may not be willing to share them with the server. Furthermore, sharing massive data sets can result in a substantial burden on the communication links between the edge devices and the server. To allow centralized training without data sharing, Federated learning (FL) was proposed in [3] as a method combining distributed training with central aggregation. This novel method of learning has been the focus of growing research attention over the last few years [4]. FL exploits the increased computational capabilities of modern edge devices to train a model on the users' side, while the server orchestrates these local training procedures and, in addition, periodically synchronizes the local models into a global one.

FL is trained by an iterative process [5]. In particular, at each FL iteration, the edge devices train a local model using their (possibly) private data, and transmit the updated model to the central server. The server aggregates the received updates into a single global model, and sends its parameters back to the edge devices [6]. Therefore, to implement FL, edge devices only need to exchange their trained model parameters, which avoids the need to share their data, thereby preserving privacy. However, the repeated exchange of updated models between the users and the server given the large number of model parameters, involves massive transmissions over throughput-limited communication channels. This challenge is particularly relevant for FL carried out over wireless networks, e.g., when the users are wireless edge devices. In addition to overloading the communication infrastructure, these repeated transmissions imply that the time required to tune the global model not only depends on the number of training iterations, but also depends on the delay induced by transmitting the model updates at each FL iteration [7]. Hence, this communication bottleneck may affect the training

time of global models trained via FL, which in turn may degrade their resulting accuracy. This motivates the design of schemes whose purpose is to limit the communication overhead due to the repeated transmissions of updated model parameters in the distributed training procedure.

Various methods have been proposed in the literature to tackle the communication bottleneck induced by the repeated model updates in FL. The works [8–17] focused on FL over wireless channels, and reduced the communication by optimizing the allocation of the channel resources, e.g., bandwidth, among the participating users, as well as limiting the amount of participating devices while scheduling when each user takes part in the overall training procedure. An additional related strategy treats the model aggregation in FL as a form of over-the-air computation [18–20]. Here, the users exploit the full resources of the wireless channel to convey their model updates at high throughput, and the resulting interference is exploited as part of the aggregation stage at the server side. These communication-oriented strategies are designed for scenarios in which the participating users communicate over the same wireless media, and are thus concerned with the division of the channel resources among the users.

An alternative approach to reduce the communication overhead, which holds also when the users do not share the same wireless channel, is to reduce the volume of the model updates conveyed at each FL iteration. Such strategies do not focus on the communication channel and how to transmit over it, but rather on what is being transmitted. As a result, they can commonly be combined with the aforementioned communication-oriented strategies. One way to limit the volume of conveyed parameters is to have each user transmit only part of its model updates, i.e., implement dimensionality reduction by sparsifying or subsampling [6, 21–25]. An alternative approach is to discretize the model updates, such that each parameter is expressed using a small number of bits, as proposed in [26–31]. More generally, the compression of the model updates can be viewed as the conversion of a high-dimensional vector, whose entries take continuous values, into a set of bits communicated over the wireless channel. Such formulations are commonly studied in the fields of quantization theory and source coding. This motivates the formulation of such compression methods for FL from a quantization perspective, which is the purpose of this chapter.

The goal of this chapter is to present a unified FL framework utilizing quantization theory, which generalizes many of the previously proposed FL compression methods. The purpose of the unified framework is to facilitate the comparison and the understanding of the differences between existing schemes in a systematic manner, as well as identify quantization theoretic tools that are particularly relevant for FL. We first introduce the basic concepts of FL and quantization theory in Section 1.2. We conclude this section with identifying the unique requirements and characteristics of FL, which affect the design of compression and quantization methods. Based on these requirements, we present in Section 1.3 quantization theory tools that are relevant for the problem at hand in a gradual and systemic manner: We begin with the basic concept of scalar quantization

and identify the need for a probabilistic design. Then, we introduce the notion of subtractive dithering as a means of reducing the distortion induced by discretization, and explain why it is specifically relevant in FL scenarios, as it can be established by having a random seed shared by the server and each user. Next, we discuss how the distortion can be further reduced by jointly discretizing multiple parameters, i.e., switching from scalar to vector quantization, in a universal fashion, without violating the basic requirements and constraints of FL. Finally, we demonstrate how quantization can be combined with lossy source coding, which provides further performance benefits from the underlying non-uniformity and sparsity of the digital representations. The resulting compression method combining all of these aforementioned quantization concepts is referred to as universal vector quantization for federated learning (UVEQFed). In Section 1.4 we analyze the performance measures of UVEQFed, including the resulting distortion induced by quantization, as well as the convergence profile of models trained using UVEQFed combined with conventional federated averaging. Our performance analysis begins by considering the distributed learning of a model using a smooth convex objective measure. The analysis demonstrates that proper usage of quantization tools result in achieving a similar asymptotic convergence profile as that of FL with uncompressed model updates, i.e., without communication constraints. Next, we present an experimental study which evaluates the performance of such quantized FL models when training neural networks with non-synthetic data sets. These numerical results illustrate the added value of each of the studied quantization tools, demonstrating that both subtractive dithering and universal vector quantizers achieves more accurate recovery of model updates in each FL iteration for the same number of bits. Furthermore, the reduced distortion is translated into improved convergence with the MNIST and CIFAR-10 data sets. We conclude this chapter with a summary of the unified UVEQFed framework and its performance in Section 1.5.

## 1.2 Preliminaries and System Model

Here we formulate the system model of quantized FL. We first review some basics in quantization theory in Section 1.2.1. Then, we formulate the conventional FL setup in Section 1.2.2, which operates without the need to quantize the model updates. Finally, we show how the throughput constraints of uplink wireless communications, i.e., the transmissions from the users to the server, gives rise to the need for quantized FL, which is formulated in Section 1.2.3.

### 1.2.1 Preliminaries in Quantization Theory

We begin by briefly reviewing the standard quantization setup, and state the definition of a quantizer:

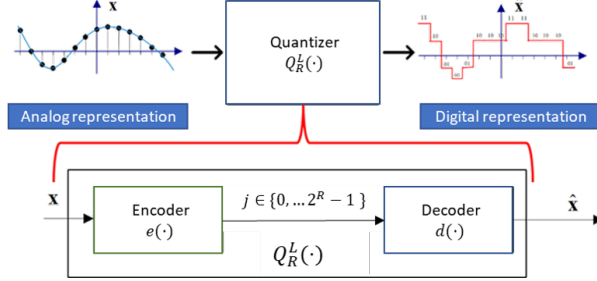


Figure 1.1 Quantization operation illustration.

**DEFINITION 1.1 (Quantizer)** A quantizer  $Q_R^L(\cdot)$  with  $R$  bits, input size  $L$ , input alphabet  $\mathcal{X}$ , and output alphabet  $\hat{\mathcal{X}}$ , consists of:

- 1) An encoder  $e : \mathcal{X}^L \mapsto \{0, \dots, 2^R - 1\} \triangleq \mathcal{U}$  which maps the input into a discrete index.
- 2) A decoder  $d : \mathcal{U} \mapsto \hat{\mathcal{X}}^L$  which maps each  $j \in \mathcal{U}$  into a codeword  $\mathbf{q}_j \in \hat{\mathcal{X}}^L$ .

We write the output of the quantizer with input  $\mathbf{x} \in \mathcal{X}^L$  as

$$\hat{\mathbf{x}} = d(e(\mathbf{x})) \triangleq Q_R^L(\mathbf{x}). \quad (1.1)$$

*Scalar quantizers* operate on a scalar input, i.e.,  $L = 1$  and  $\mathcal{X}$  is a scalar space, while *vector quantizers* have a multivariate input. An illustration of a quantization system is depicted in Fig. 1.1.

The basic problem in quantization theory is to design a  $Q_R^L(\cdot)$  quantizer in order to minimize some distortion measure  $\delta : \mathcal{X}^L \times \hat{\mathcal{X}}^L \mapsto \mathcal{R}^+$  between its input and its output. The performance of a quantizer is characterized using its quantization rate  $\frac{R}{L}$ , and the expected distortion  $\mathbb{E}\{\delta(\mathbf{x}, \hat{\mathbf{x}})\}$ . A common distortion measure is the mean-squared error (MSE), i.e.,  $\delta(\mathbf{x}, \hat{\mathbf{x}}) \triangleq \|\mathbf{x} - \hat{\mathbf{x}}\|^2$ .

Characterizing the optimal quantizer, i.e., the one which minimizes the distortion for a given quantization rate, and its trade-off between distortion and rate is in general a very difficult task. Optimal quantizers are thus typically studied assuming either high quantization rate, i.e.,  $\frac{R}{m} \rightarrow \infty$ , see, e.g., [32], or asymptotically large inputs, namely,  $L \rightarrow \infty$ , via rate-distortion theory [33, Ch. 10]. One of the fundamental results in quantization theory is that vector quantizers are superior to scalar quantizers in terms of their rate-distortion tradeoff. For example, for large quantization rate, even for i.i.d. inputs, vector quantization outperforms scalar quantization, with a distortion gap of 4.35 dB for Gaussian inputs with the MSE distortion [34, Ch. 23.2].

### 1.2.2 Preliminaries in FL

#### FL System Model

In this section we describe the conventional FL framework proposed in [3]. Here, a centralized server is training a model consisting of  $m$  parameters based on

labeled samples available at a set of  $K$  remote users. The model is trained to minimize a loss function  $\ell(\cdot; \cdot)$ . Letting  $\{\mathbf{x}_i^{(k)}, \mathbf{y}_i^{(k)}\}_{i=1}^{n_k}$  be the set of  $n_k$  labeled training samples available at the  $k$ th user,  $k \in \{1, \dots, K\} \triangleq \mathcal{K}$ , FL aims at recovering the  $m \times 1$  weights vector  $\mathbf{w}^\circ$  satisfying

$$\mathbf{w}^\circ = \arg \min_{\mathbf{w}} \left\{ F(\mathbf{w}) \triangleq \sum_{k=1}^K \alpha_k F_k(\mathbf{w}) \right\}. \quad (1.2)$$

Here, the weighting average coefficients  $\{\alpha_k\}$  are non-negative satisfying  $\sum \alpha_k = 1$ , and the local objective functions are defined as the empirical average over the corresponding training set, i.e.,

$$F_k(\mathbf{w}) \triangleq F_k(\mathbf{w}; \{\mathbf{x}_i^{(k)}, \mathbf{y}_i^{(k)}\}_{i=1}^{n_k}) \triangleq \frac{1}{n_k} \sum_{i=1}^{n_k} \ell(\mathbf{w}; (\mathbf{x}_i^{(k)}, \mathbf{y}_i^{(k)})). \quad (1.3)$$

### Federated Averaging

*Federated averaging* [3] aims at recovering  $\mathbf{w}^\circ$  using iterative subsequent updates. In each update of time instance  $t$ , the server shares its current model, represented by the vector  $\mathbf{w}_t \in \mathcal{R}^m$ , with the users. The  $k$ th user,  $k \in \mathcal{K}$ , uses its set of  $n_k$  labeled training samples to retrain the model  $\mathbf{w}_t$  over  $\tau$  time instances into an updated model  $\tilde{\mathbf{w}}_{t+\tau}^{(k)} \in \mathcal{R}^m$ . Commonly,  $\tilde{\mathbf{w}}_{t+\tau}^{(k)}$  is obtained by  $\tau$  stochastic gradient descent (SGD) steps applied to  $\mathbf{w}_t$ , executed over the local data set, i.e.,

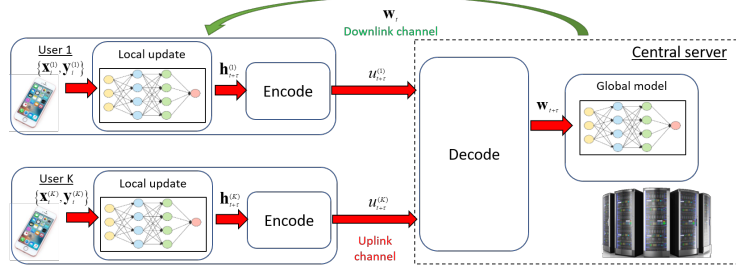
$$\tilde{\mathbf{w}}_{t+1}^{(k)} = \tilde{\mathbf{w}}_t^{(k)} - \eta_t \nabla F_k^{i_t^{(k)}}(\tilde{\mathbf{w}}_t^{(k)}), \quad (1.4)$$

where  $i_t^{(k)}$  is a sample index chosen uniformly from the local data of the  $k$ th user at time  $t$ . When the local updates are carried out via (1.4), federated averaging specializes the local SGD method [35], and the terms are often used interchangeably in the literature.

Having updated the model weights, the  $k$ th user conveys its model update, denoted as  $\mathbf{h}_{t+\tau}^{(k)} \triangleq \tilde{\mathbf{w}}_{t+\tau}^{(k)} - \mathbf{w}_t$ , to the server. The server synchronizes the global model by averaging the model updates via

$$\mathbf{w}_{t+\tau} = \mathbf{w}_t + \sum_{k=1}^K \alpha_k \mathbf{h}_{t+\tau}^{(k)} = \sum_{k=1}^K \alpha_k \tilde{\mathbf{w}}_{t+\tau}^{(k)}. \quad (1.5)$$

By repeating this procedure over multiple iterations, the resulting global model can be shown to converge to  $\mathbf{w}^\circ$  under various objective functions [35–37]. The number of local SGD iterations can be any positive integer. For  $\tau = 1$ , the model updates  $\{\mathbf{h}_{t+\tau}^{(k)}\}$  are the scaled stochastic gradients, and thus the local SGD method effectively implements mini-batch SGD. While such a setting results in a learning scheme that is simpler to analyze and is less sensitive to data heterogeneity compared to using large values of  $\tau$ , it requires much more communications between the participating entities [38], and may give rise to privacy concerns [39].



**Figure 1.2** Federated learning with bit rate constraints.

### 1.2.3 FL with Quantization Constraints

The federated averaging method relies on the ability of the users to repeatedly convey their model updates to the server without errors. This implicitly requires the users and the server to communicate over ideal links of infinite throughput. Since upload speeds are typically more limited compared to download speeds [40], the users need to communicate a finite-bit quantized representation of their model update, resulting in the quantized FL setup whose model we now describe.

#### Quantized FL System Model

The number of model parameters  $m$  can be very large, particularly when training highly-parameterized deep neural networks (DNNs). The requirement to limit the volume of data conveyed over the uplink channel implies that the model updates should be quantized prior to its transmission. Following the formulation of the quantization problem in Section 1.2.1, this implies that each user should encode the model update into a digital representation consisting of a finite number of bits (a codeword), and the server has to decode each of these codewords describing the local model into a global model update. The  $k$ th model update  $\mathbf{h}_{t+\tau}^{(k)}$  is therefore encoded into a digital codeword of  $R_k$  bits denoted as  $u_t^{(k)} \in \{0, \dots, 2^{R_k} - 1\} \triangleq \mathcal{U}_k$ , using an encoding function whose input is  $\mathbf{h}_{t+\tau}^{(k)}$ , i.e.,

$$e_{t+\tau}^{(k)} : \mathcal{R}^m \mapsto \mathcal{U}_k. \quad (1.6)$$

The uplink channel is modeled as a bit-constrained link whose transmission rate does not exceed Shannon capacity, i.e., each  $R_k$  bit codeword is recovered by the server without errors, as commonly assumed in the FL literature [6, 21–23, 25–31, 41]. The server uses the received codewords  $\{u_{t+\tau}^{(k)}\}_{k=1}^K$  to reconstruct  $\hat{\mathbf{h}}_{t+\tau} \in \mathcal{R}^m$ , obtained via a joint decoding function

$$d_{t+\tau} : \mathcal{U}_1 \times \dots \times \mathcal{U}_K \mapsto \mathcal{R}^m. \quad (1.7)$$

The recovered  $\hat{\mathbf{h}}_{t+\tau}$  is an estimate of the weighted average  $\sum_{k=1}^K \alpha_k \mathbf{h}_{t+\tau}^{(k)}$ . Finally, the global model  $\mathbf{w}_{t+\tau}$  is updated via

$$\mathbf{w}_{t+\tau} = \mathbf{w}_t + \hat{\mathbf{h}}_{t+\tau}. \quad (1.8)$$

An illustration of this FL procedure is depicted in Fig. 1.2. Clearly, if the number of allowed bits is sufficiently large, the distance  $\|\hat{\mathbf{h}}_{t+\tau} - \sum_{k=1}^K \alpha_k \mathbf{h}_{t+\tau}^{(k)}\|^2$  can be made arbitrarily small, allowing the server to update the global model as the desired weighted average (1.5)

In the presence of a limited bit budget, i.e., small values of  $\{R_k\}$ , distortion is induced, which can severely degrade the ability of the server to update its model. This motivates the need for efficient quantization methods for FL, which are to be designed under the unique requirements of FL that we now describe.

### Quantization Requirements

Tackling the design of quantized FL from a purely information theoretic perspective, i.e., as a lossy source code [33, Ch. 10], inevitably results in utilizing complex vector quantizers. In particular, in order to approach the optimal trade-off between number of bits and quantization distortion, one must jointly map  $L$  samples together, where  $L$  should be an arbitrarily large number, by creating a partition of the  $L$ -dimensional hyperspace which depends on the distribution of the model updates. Furthermore, the distributed nature of the FL setups implies that the reconstruction accuracy can be further improved by utilizing infinite-dimensional vector quantization as part of a distributed lossy source coding scheme [42, Ch. 11], such as Wyner-Ziv coding [43]. However, these coding techniques tend to be computationally complex, and require each of the nodes participating in the procedure to have accurate knowledge of the joint distribution of the complete set of continuous-amplitude values to be discretized, which is not likely to be available in practice.

Therefore, to study quantization schemes while faithfully representing FL setups, one has to account for the following requirements and assumptions:

- A1. All users share the same encoding function, denoted as  $e_t^{(k)}(\cdot) = e_t(\cdot)$  for each  $k \in \mathcal{K}$ . This requirement, which was also considered in [6], significantly simplifies FL implementation.
- A2. No *a-priori* knowledge or distribution of  $\mathbf{h}_{t+\tau}^{(k)}$  is assumed.
- A3. As in [6], the users and the server share a source of common randomness. This is achieved by, e.g., letting the server share with each user a random seed along with the weights. Once a different seed is conveyed to each user, it can be used to obtain a dedicated source of common randomness shared by the server and each of the users for the entire FL procedure.

The above requirements, and particularly A1-A2, are stated to ensure feasibility of the quantization scheme for FL applications. Ideally, a compression mechanism should exploit knowledge about the distribution of its input, and different distributions would yield different encoding mechanisms. However, the fact that prior statistical knowledge about the distribution of the model updates is likely to be unavailable, particularly when training deep models, while it is desirable to design a single encoding mechanism that can be used by all devices, motivates the statement of requirements A1-A2. In fact, these conditions give

rise to the need for a *universal quantization* approach, namely, a scheme that operates reliably regardless of the distribution of the model updates and without prior knowledge of this distribution.

### 1.3 Quantization for FL

Next, we detail various mechanisms for quantizing the model updates conveyed over the uplink channel in FL. We begin with the common method of probabilistic scalar quantization in Section 1.3.1. Then in Sections 1.3.2-1.3.3 we show how distortion can be reduced while accounting for the FL requirements A1-A3 by introducing subtractive dithering and vector quantization, respectively. Finally, we present a unified formulation for quantized FL based on the aforementioned techniques combined with lossless source coding and overload prevention in Section 1.3.4.

#### 1.3.1 Probabilistic Scalar Quantization

The most simple and straight-forward approach to discretize the model updates is to utilize quantizers. Here, a scalar quantizer  $Q_R^1(\cdot)$  is set to some fixed partition of the real line, and each user encodes its model update  $\mathbf{h}_{t+\tau}^{(k)}$  by applying  $Q_R^1(\cdot)$  to it entry-wise. Arguably the most common scalar quantization rule is the uniform mapping, which for a given support  $\gamma > 0$  and quantization step size  $\Delta = \frac{2\gamma}{2^R}$  is given by

$$Q_R^1(x) = \begin{cases} \Delta \left( \lfloor \frac{x}{\Delta} \rfloor + \frac{1}{2} \right), & \text{for } |x| < \gamma \\ \text{sign}(x) \left( \gamma - \frac{\Delta}{2} \right), & \text{else,} \end{cases} \quad (1.9)$$

where  $\lfloor \cdot \rfloor$  denotes rounding to the next smaller integer and  $\text{sign}(\cdot)$  is the signum function. The overall number of bits used for representing the model update is thus  $mR$ , which adjusts the resolution of the quantizers to yield a total of  $R_k$  bits to describe the model update. The server then uses the quantized model update to compute the aggregated global model by averaging these digital representations into (1.8). In its most coarse form, the quantizer represents each entry using a single bit, as in, e.g., signSGD [26, 31]. An illustration of this simplistic quantization scheme is given in Fig. 1.3.

The main drawback of the scalar quantization described in (1.9) follows from the fact that its distortion is a deterministic function of its input. To see this, consider for example the case in which the quantizer  $Q_R^1(\cdot)$  implements rounding to the nearest integer, and all the users compute the first entry of their model updates as 1.51. In such a case, all the users will encode this entry as the integer value 2, and thus the first entry of  $\hat{\mathbf{h}}_{t+\tau}$  in (1.8) also equals 2, resulting in a possibly notable error in the aggregated model. This motivates the usage of *probabilistic quantization*, where the distortion is a random quantity. Considering again the previous example, if instead of having each user encode 1.51 into 2,



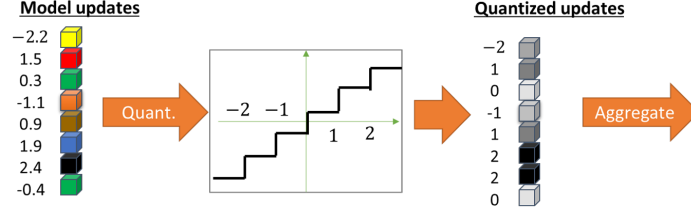


Figure 1.3 Model scalar quantization illustration.

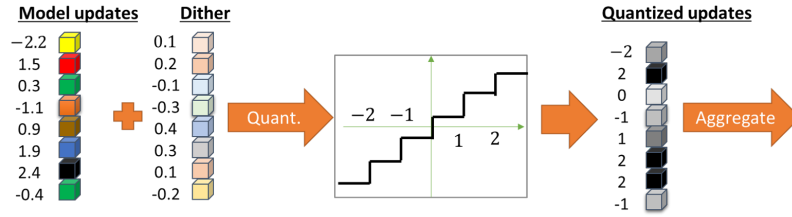
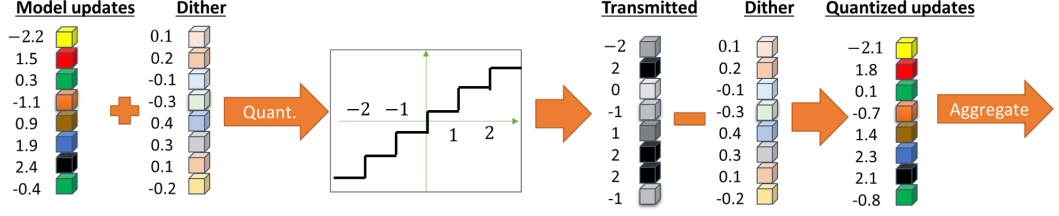


Figure 1.4 Model probabilistic scalar quantization illustration.

the users would have 51% probability of encoding it 2, and 49% probability of encoding it as 1. The aggregated model is expected to converge to the desired update value of 1.51 as the number of users  $K$  grows by the law of large numbers.

Various forms of probabilistic quantization for distributed learning have been proposed in the literature [41]. These include using one-bit sign quantizers [26], ternary quantization [27], uniform quantization [28], and non-uniform quantization [29]. Probabilistic scalar quantization can be treated as a form of *dithered quantization* [44, 45], where the continuous-amplitude quantity is corrupted by some additive noise, referred to as dither signal, which is typically uniformly distributed over its corresponding decision region. Consequently, the quantized updates of the  $k$ th user are given by applying  $Q_R^1(\cdot)$  to  $\mathbf{h}_{t+\tau}^{(k)} + \mathbf{z}_{t+\tau}^{(k)}$  element-wise, where  $\mathbf{z}_{t+\tau}^{(k)}$  denotes the dither signal. When the quantization mapping consists of the uniform partition in (1.9), as used in the QSGD algorithm [28], this operation specializes non-subtractive dithered quantization [45], and the entries of the dither signal  $\mathbf{z}_{t+\tau}^{(k)}$  are typically i.i.d. and uniformly distributed over  $[-\Delta/2, \Delta/2]$  [45]. An illustration of this continuous-to-discrete mapping is depicted in Fig. 1.4

Probabilistic quantization overcomes errors of a deterministic nature, as discussed above in the context of conventional scalar quantization. However, the addition of the dither signal also increases the distortion in representing each model update in discrete form [46]. In particular, while probabilistic quantization reduces the effect of the distortion induced on the aggregated  $\hat{\mathbf{h}}_{t+\tau}$  in (1.8) compared to conventional scalar quantization, it results in the discrete representation of each individual update  $\mathbf{h}_{t+\tau}^{(k)}$  being less accurate. This behavior is



**Figure 1.5** Model subtractive dithered scalar quantization illustration.

also observed when comparing the quantized updates in Figs. 1.3 and 1.4. This excess distortion can be reduced by utilizing subtractive dithering strategies, as detailed in the following section.

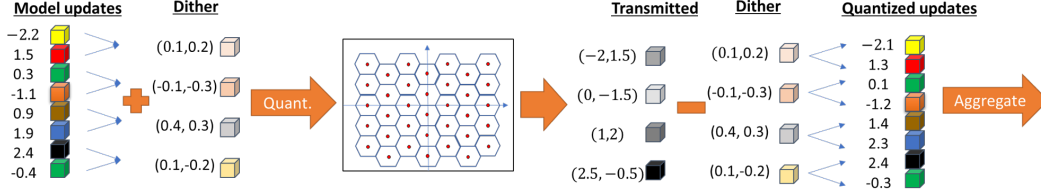
### 1.3.2 Subtractive Dithered Scalar Quantization

Subtractive dithered quantization extends probabilistic quantization by introducing an additional decoding step, rather than directly using the discrete code-words as the compressed digital representation. The fact that each of the users can share a source of local randomness with the server by assumption A3 implies that the server can generate the realization of the dither signal  $\mathbf{z}_{t+\tau}^{(k)}$ . Consequently, instead of using the element-wise quantized version of  $\mathbf{h}_{t+\tau}^{(k)} + \mathbf{z}_{t+\tau}^{(k)}$  received from the  $k$ th user, denoted here as  $Q(\mathbf{h}_{t+\tau}^{(k)} + \mathbf{z}_{t+\tau}^{(k)})$ , the user sets its representation of  $\mathbf{h}_{t+\tau}^{(k)}$  to be  $Q(\mathbf{h}_{t+\tau}^{(k)} + \mathbf{z}_{t+\tau}^{(k)}) - \mathbf{z}_{t+\tau}^{(k)}$ . An illustration of this procedure is depicted in Fig. 1.5.

The subtraction of the dither signal upon decoding reduces its excess distortion in a manner that does not depend on the distribution of the continuous-amplitude value [47]. As such, it is also referred to as *universal scalar quantization*. In particular, for uniform quantizers of the form (1.9) where the input lies within the support  $[-\gamma, \gamma]$ , it holds that the effect of the quantization can be rigorously modeled as an additive noise term whose entries are uniformly distributed over  $[-\Delta/2, \Delta/2]$ , regardless of the values of the realization and the distribution of the model updates  $\mathbf{h}_{t+\tau}^{(k)}$  [45]. This characterization implies that the excess distortion due to dithering is mitigated for each model update individually, while the overall distortion is further reduced in aggregation, as federated averaging results in this additive noise term effectively approaching its mean value of zero by the law of large numbers.

### 1.3.3 Subtractive Dithered Vector Quantization

While scalar quantizers are simple to implement, they process each sample of the model updates using the same continuous-to-discrete mapping. Consequently, scalar quantizers are known to be inferior to vector quantizers, which jointly



**Figure 1.6** Model subtractive dithered vector quantization illustration.

map a set of  $L > 1$  samples into a single digital codeword, in terms of their achievable distortion for a given number of bits. In this section we detail how the concept of subtractive dithered quantization discussed in the previous section can be extended to vector quantizers, as illustrated in Fig. 1.6. The extension of universal quantization via subtractive dithering to multivariate samples reviewed here is based on lattice quantization [48]. To formulate the notion of such dithered vector quantizers, we first briefly review lattice quantization, after which we discuss its usage for FL uplink compression.

### Lattice Quantization

Let  $L$  be a fixed positive integer, referred to henceforth as the lattice dimension, and let  $\mathbf{G}$  be a non-singular  $L \times L$  matrix, which denotes the lattice generator matrix. For simplicity, we assume that  $M \triangleq \frac{m}{L}$  is an integer, where  $m$  is the number of model parameters, although the scheme can also be applied when this does not hold by replacing  $M$  with  $\lceil M \rceil$ . Next, we use  $\mathcal{L}$  to denote the lattice, which is the set of points in  $\mathcal{R}^L$  that can be written as an integer linear combination of the columns of  $\mathbf{G}$ , i.e.,

$$\mathcal{L} \triangleq \{\mathbf{x} = \mathbf{G}\mathbf{l} : \mathbf{l} \in \mathbb{Z}^L\}. \quad (1.10)$$

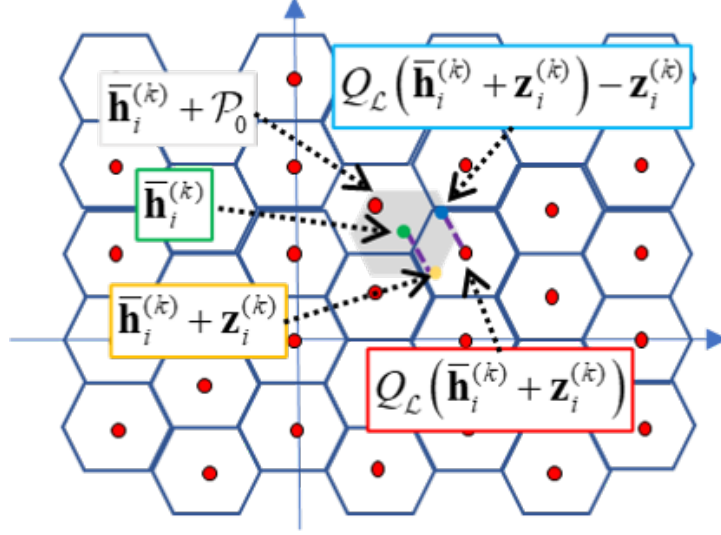
A lattice quantizer  $Q_{\mathcal{L}}(\cdot)$  maps each  $\mathbf{x} \in \mathcal{R}^L$  to its nearest lattice point, i.e.,  $Q_{\mathcal{L}}(\mathbf{x}) = \mathbf{l}_{\mathbf{x}}$  where  $\mathbf{l}_{\mathbf{x}} \in \mathcal{L}$  if  $\|\mathbf{x} - \mathbf{l}_{\mathbf{x}}\| \leq \|\mathbf{x} - \mathbf{l}\|$  for every  $\mathbf{l} \in \mathcal{L}$ . Finally, let  $\mathcal{P}_0$  be the basic lattice cell [49], i.e., the set of points in  $\mathcal{R}^L$  which are closer to  $\mathbf{0}$  than to any other lattice point:

$$\mathcal{P}_0 \triangleq \{\mathbf{x} \in \mathcal{R}^L : \|\mathbf{x}\| < \|\mathbf{x} - \mathbf{p}\|, \forall \mathbf{p} \in \mathcal{L}/\{\mathbf{0}\}\}. \quad (1.11)$$

For example, when  $\mathbf{G} = \Delta \cdot \mathbf{I}_L$  for some  $\Delta > 0$ , then  $\mathcal{L}$  is the square lattice, for which  $\mathcal{P}_0$  is the set of vectors  $\mathbf{x} \in \mathcal{R}^L$  whose  $\ell_{\infty}$  norm is not larger than  $\frac{\Delta}{2}$ . For this setting,  $Q_{\mathcal{L}}(\cdot)$  implements entry-wise scalar uniform quantization with spacing  $\Delta$  [34, Ch. 23]. This is also the case when  $L = 1$ , for which  $Q_{\mathcal{L}}(\cdot)$  specializes scalar uniform quantization with spacing dictated by the (scalar)  $\mathbf{G}$ .

### Subtractive Dithered Lattice Quantization

Lattice quantizers can be exploited to realize universal vector quantization. Such mappings jointly quantize multiple samples, thus benefiting from the improved



**Figure 1.7** Subtractive dithered lattice quantization illustration.

rate-distortion tradeoff of vector quantizers, while operating in a manner which is invariant of the distribution of the continuous-amplitude inputs [48].

To formulate this operation, we note that in order to apply an  $L$ -dimensional lattice quantizer to  $\mathbf{h}_{t+\tau}^{(k)}$ , it must first be divided into  $M$  distinct  $L \times 1$  vectors, denoted  $\{\bar{\mathbf{h}}_i^{(k)}\}_{i=1}^M$ . In order to quantize each sub-vector  $\bar{\mathbf{h}}_i^{(k)}$ , it is first corrupted with a dither vector  $\mathbf{z}_i^{(k)}$ , which is here uniformly distributed over the basic lattice cell  $\mathcal{P}_0$ , and then quantized using a lattice quantizer  $\mathcal{Q}_{\mathcal{L}}(\cdot)$ . On the decoder side, the representation of  $\bar{\mathbf{h}}_i^{(k)}$  is obtained by subtracting  $\mathbf{z}_i^{(k)}$  from the discrete  $\mathcal{Q}_{\mathcal{L}}(\bar{\mathbf{h}}_i^{(k)} + \mathbf{z}_i^{(k)})$ . An example of this procedure for  $L = 2$  is illustrated in Fig. 1.7.

#### 1.3.4 Unified Formulation

The aforementioned strategies give rise to a unified framework for model updates quantization in FL. Before formulating the resulting encoding and decoding mappings in a systematic manner, we note that two additional considerations of quantization and compression should be accounted for in the formulation. These are the need to prevent overloading of the quantizers, and the ability to further compress the discretized representations via lossless source coding.

##### Overloading Prevention

Quantizers are typically required to operate within their dynamic range, namely, when the input lies in the support of the quantizer. For uniform scalar quantizers as in (1.9), this implies that the magnitude of the input must not be larger than  $\gamma$ . The same holds for multivariate lattice quantizers; as an infinite number of

lattice regions are required to span  $\mathcal{R}^L$ , the input must be known to lie in some  $L$ -dimensional ball in order to utilize a finite number of lattice points, which in turn implies a finite number of different discrete representations. In particular, the desired statistical properties which arise from the usage of dithering, i.e., that the distortion is uncorrelated with the input and can thus be reduced by averaging, hold when the input is known to lie within the quantizer support [45].

In order to guarantee that the quantizer is not overloaded, namely, that each continuous-amplitude value lies in the quantizer support, the model updates vector  $\mathbf{h}_{t+\tau}^{(k)}$  can be scaled by  $\zeta \|\mathbf{h}_{t+\tau}^{(k)}\|$  for some parameter  $\zeta > 0$ . This setting guarantees that the elements  $\{\bar{\mathbf{h}}_i^{(k)}\}_{i=1}^M$  of which the compressed  $\mathbf{h}_{t+\tau}^{(k)}$  is comprised all reside inside the  $L$ -dimensional ball with radius  $\zeta^{-1}$ . The number of lattice points is not larger than  $\frac{\pi^{L/2}}{\zeta^L \Gamma(1+L/2) \det(\mathbf{G})}$  [50, Ch. 2], where  $\Gamma(\cdot)$  is the Gamma function. Note that the scalar quantity  $\zeta \|\mathbf{h}_{t+\tau}^{(k)}\|$  depends on the vector  $\mathbf{h}_{t+\tau}^{(k)}$ , and must thus be quantized with high resolution and conveyed to the server, to be accounted for in the decoding process. The overhead in accurately quantizing the single scalar quantity  $\zeta \|\mathbf{h}_{t+\tau}^{(k)}\|$  is typically negligible compared to the number of bits required to convey the set of vectors  $\{\bar{\mathbf{h}}_i^{(k)}\}_{i=1}^M$ , hardly affecting the overall quantization rate.

### Lossless Source Coding

As defined in Section 1.2.1, quantizers can output a finite number of different codewords. The amount of codewords dictates the number of bits used  $R$ , as each codeword can be mapped into a different combination of  $R$  bits, and conveyed in digital form over the rate-constrained uplink channel. However, these digital representations are in general not uniformly distributed. In particular, model updates are often approximately sparse, which is the property exploited in sparsification-based compression schemes [21, 23]. Consequently, the codeword corresponding to (almost) zero values is likely to be assigned more often than other codewords by the quantizer.

This property implies that the discrete output of the quantizer, i.e., the vector  $Q(\mathbf{h}_{t+\tau}^{(k)} + \mathbf{z}_{t+\tau}^{(k)})$ , can be further compressed by lossless source coding. Various lossless source coding schemes, including arithmetic, Lempel-Ziv, and Elias codes, are capable of compressing a vector of discrete symbols into an amount of bits approaching the most compressed representation, dictated by the entropy of the vector [33, Ch. 13]. When the distribution of the discrete vector  $Q(\mathbf{h}_{t+\tau}^{(k)} + \mathbf{z}_{t+\tau}^{(k)})$  is notably different from being uniform, as is commonly the case in quantized FL, the incorporation of such entropy coding can substantially reduce the number of bits in a lossless manner, i.e., without inducing additional distortion.

### Encoder-Decoder Formulation

Based on the above considerations of overloading prevention and the potential of entropy coding, we now formulate a unified quantized FL strategy coined UVeQFed. UVeQFed generalizes the quantization strategies for FL detailed in

Sections 1.3.1-1.3.3, where the usage of scalar versus vector quantizers is dictated by the selection of the dimension parameter  $L$ : When  $L = 1$  UVeQFed implements scalar quantization, while with  $L > 1$  it results in subtractive dithered lattice quantization. Specifically, UVeQFed consists of the following encoding and decoding mappings:

**Encoder:** The encoding function  $e_{t+\tau}(\cdot)$  (1.6) for each user includes the following steps:

- E1. **Normalize and partition:** The  $k$ th user scales  $\mathbf{h}_{t+\tau}^{(k)}$  by  $\zeta \|\mathbf{h}_{t+\tau}^{(k)}\|$  for some  $\zeta > 0$ , and divides the result into  $M$  distinct  $L \times 1$  vectors, denoted  $\{\bar{\mathbf{h}}_i^{(k)}\}_{i=1}^M$ . The scalar quantity  $\zeta \|\mathbf{h}_{t+\tau}^{(k)}\|$  is quantized separately from  $\{\bar{\mathbf{h}}_i^{(k)}\}_{i=1}^M$  with high resolution, e.g., using a uniform scalar quantizer with at least 12 bits, such that it can be recovered at the decoder with negligible distortion.
- E2. **Dithering:** The encoder utilizes the source of common randomness, e.g., a shared seed, to generate the set of  $L \times 1$  dither vectors  $\{\mathbf{z}_i^{(k)}\}_{i=1}^M$ , which are randomized in an i.i.d. fashion, independently of  $\mathbf{h}_{t+\tau}^{(k)}$ , from a uniform distribution over  $\mathcal{P}_0$ .
- E3. **Quantization:** The vectors  $\{\bar{\mathbf{h}}_i^{(k)}\}_{i=1}^M$  are discretized by adding the dither vectors and applying lattice/uniform quantization, namely, by computing  $\{Q_{\mathcal{L}}(\bar{\mathbf{h}}_i^{(k)} + \mathbf{z}_i^{(k)})\}$ .
- E4. **Entropy coding:** The discrete values  $\{Q_{\mathcal{L}}(\bar{\mathbf{h}}_i^{(k)} + \mathbf{z}_i^{(k)})\}$  are encoded into a digital codeword  $u_{t+\tau}^{(k)}$  in a lossless manner.

**Decoder:** The decoding mapping  $d_{t+\tau}(\cdot)$  implements the following:

- D1. **Entropy decoding:** The server first decodes each digital codeword  $u_{t+\tau}^{(k)}$  into the discrete value  $\{Q_{\mathcal{L}}(\bar{\mathbf{h}}_i^{(k)} + \mathbf{z}_i^{(k)})\}$ . Since the encoding is carried out using a lossless source code, the discrete values are recovered without any errors.
- D2. **Dither subtraction:** Using the source of common randomness, the server generates the dither vectors  $\{\mathbf{z}_i^{(k)}\}$ , which can be carried out rapidly and at low complexity using random number generators as the dither vectors obey a uniform distribution. The server then subtracts the corresponding vector by computing  $\{Q_{\mathcal{L}}(\bar{\mathbf{h}}_i^{(k)} + \mathbf{z}_i^{(k)}) - \mathbf{z}_i^{(k)}\}$ .
- D3. **Collecting and scaling:** The values computed in the previous step are collected into an  $m \times 1$  vector  $\hat{\mathbf{h}}_{t+\tau}^{(k)}$  using the inverse operation of the partitioning and normalization in Step E1.
- D4. **Model recovery:** The recovered matrices are combined into an updated model based on (1.8). Namely,

$$\mathbf{w}_{t+\tau} = \mathbf{w}_t + \sum_{k=1}^K \alpha_k \hat{\mathbf{h}}_{t+\tau}^{(k)}. \quad (1.12)$$

A block diagram of the proposed scheme is depicted in Fig. 1.8. The use of subtractive dithered quantization in Steps E2-E3 and D2 allow obtaining

a digital representation which is relatively close to the true quantity, without relying on prior knowledge of its distribution. For non-subtractive dithering, as in conventional probabilistic quantization such as QSGD [28], one must skip the dither subtraction step D2, as illustrated in Fig. 1.8. For clarity, we use the term UVeQFed to refer to the encoder-decoder pair consisting of all the detailed steps E1-D4, i.e., with subtractive dithered quantization. The joint decoding aspect of these schemes is introduced in the final model recovery Step D4. The remaining encoding-decoding procedure, i.e., Steps E1-D3, is carried out independently for each user.

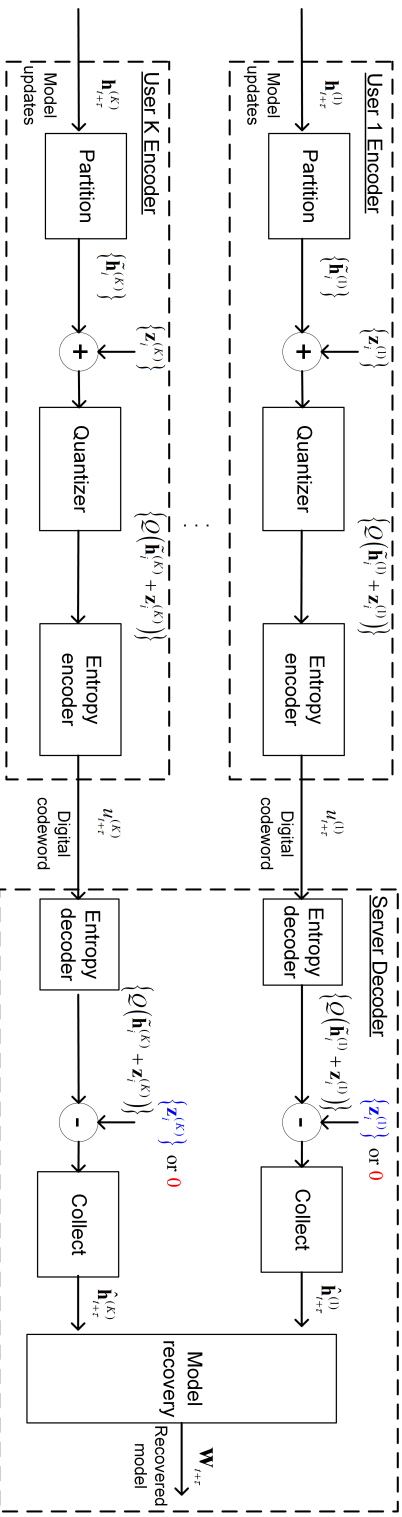
UVeQFed has several clear advantages. First, while it is based on information theoretic arguments, the resulting architecture is rather simple to implement. In particular, both subtractive dithered quantization as well as entropy coding are concrete and established methods which can be realized with relatively low complexity and feasible hardware requirements. Increasing the lattice dimension  $L$  reduces the distortion, thus leading to more improved trained models, at the cost of increased complexity in the quantization step E3. The numerical study presented in Section 1.4.2 demonstrates that the accuracy of the trained model can be notably improved by using two-dimensional lattices compared to utilizing scalar quantizers, i.e., setting  $L = 2$  instead of  $L = 1$ . The source of common randomness needed for generating the dither vectors can be obtained by sharing a common seed between the server and users, as discussed in the statement of requirement A3. The statistical characterization of the quantization error of such quantizers does not depend on the distribution of the model updates. This analytical tractability allows us to rigorously show that combining UVeQFed with federated averaging mitigates the quantization error, which we show in the following section.

## 1.4 Performance Analysis

In this section we study and compare the performance of different quantization strategies that arise from the unified formulation presented in the previous section. We first present the theoretical performance of UVeQFed in terms of its resulting distortion and FL convergence for convex objectives. Then, we numerically compare the FL performance with different quantization strategies using both synthetic and non-synthetic datasets

### 1.4.1 Theoretical Performance

Here, we theoretically characterize the performance of UVeQFed, i.e., the encoder-decoder pair detailed in Section 1.3.4. The characterization holds for both uniform scalar quantizers as well as lattice vector quantizers, depending on the setting of the parameter  $L$ .



**Figure 1.8** Unified formulation of quantized FL illustration, with subtractive dithering (blue fonts at the decoder) or without it (red fonts).



### Assumptions

We first introduce the assumptions on the objective functions  $F_k(\cdot)$  in light of which the theoretical analysis of the resulting distortion and the convergence of local SGD with UVeQFed is carried out in the sequel. In particular, our theoretical performance characterization utilizes the following assumptions:

- AS1. The expected squared  $\ell_2$  norm of the random vector  $\nabla F_k^i(\mathbf{w})$ , representing the stochastic gradient evaluated at  $\mathbf{w}$ , is bounded by some  $\xi_k^2 > 0$  for all  $\mathbf{w} \in \mathcal{R}^m$ .
- AS2. The local objective functions  $\{F_k(\cdot)\}$  are all  $\rho_s$ -smooth, namely, for all  $\mathbf{v}_1, \mathbf{v}_2 \in \mathcal{R}^m$  it holds that

$$F_k(\mathbf{v}_1) - F_k(\mathbf{v}_2) \leq (\mathbf{v}_1 - \mathbf{v}_2)^T \nabla F_k(\mathbf{v}_2) + \frac{1}{2} \rho_s \|\mathbf{v}_1 - \mathbf{v}_2\|^2.$$

- AS3. The local objective functions  $\{F_k(\cdot)\}$  are all  $\rho_c$ -strongly convex, namely, for all  $\mathbf{v}_1, \mathbf{v}_2 \in \mathcal{R}^m$  it holds that

$$F_k(\mathbf{v}_1) - F_k(\mathbf{v}_2) \geq (\mathbf{v}_1 - \mathbf{v}_2)^T \nabla F_k(\mathbf{v}_2) + \frac{1}{2} \rho_c \|\mathbf{v}_1 - \mathbf{v}_2\|^2.$$

Assumption AS1 on the stochastic gradients is often employed in distributed learning studies [35, 36, 51]. Assumptions AS2-AS3 are also commonly used in FL convergence studies [35, 36], and hold for a broad range of objective functions used in FL systems, including  $\ell_2$ -norm regularized linear regression and logistic regression [36].

### Distortion Analysis

We begin our performance analysis by characterizing the distortion induced by quantization. The need to represent the model updates  $\mathbf{h}_{t+\tau}^{(k)}$  using a finite number of bits inherently induces some distortion, i.e., the recovered vector is  $\hat{\mathbf{h}}_{t+\tau}^{(k)} = \mathbf{h}_{t+\tau}^{(k)} + \boldsymbol{\epsilon}_{t+\tau}^{(k)}$ . The error in representing  $\zeta \|\mathbf{h}_{t+\tau}^{(k)}\|$  is assumed to be negligible. For example, the normalized quantization error is on the order of  $10^{-7}$  for 12 bit quantization of a scalar value, and decreases exponentially with each additional bit [34, Ch. 23].

Let  $\bar{\sigma}_{\mathcal{L}}^2$  be the normalized second order lattice moment, defined as  $\bar{\sigma}_{\mathcal{L}}^2 \triangleq \int_{\mathcal{P}_0} \|\mathbf{x}\|^2 d\mathbf{x} / \int_{\mathcal{P}_0} d\mathbf{x}$  [52]. For uniform scalar quantizers (1.9), this quantity equals  $\Delta^2/3$ , i.e., the second-order moment of a uniform distribution over the quantizer support. The moments of the quantization error  $\boldsymbol{\epsilon}_{t+\tau}^{(k)}$  satisfy the following theorem, which is based on the properties of non-overloaded subtractive dithered quantizers [49]:

**THEOREM 1.2** *The quantization error vector  $\boldsymbol{\epsilon}_{t+\tau}^{(k)}$  has zero-mean entries and satisfies*

$$\mathbb{E}\{\|\boldsymbol{\epsilon}_{t+\tau}^{(k)}\|^2 | \mathbf{h}_{t+\tau}^{(k)}\} = \zeta^2 \|\mathbf{h}_{t+\tau}^{(k)}\|^2 M \bar{\sigma}_{\mathcal{L}}^2. \quad (1.13)$$

Theorem 1.2 characterizes the distortion in quantizing the model updates using UVeQFed. Due to the usage of vector quantizers, the dependence of the

expected error norm on the number of bits is not explicit in (1.13), but rather encapsulated in the lattice moment  $\bar{\sigma}_{\mathcal{L}}^2$ . To observe that (1.13) indeed represents lower distortion compared to previous FL quantization schemes, we note that even when scalar quantizers are used, i.e.,  $L = 1$  for which  $\frac{1}{L}\bar{\sigma}_{\mathcal{L}}^2$  is known to be largest [52], the resulting quantization is reduced by a factor of 2 compared to conventional probabilistic scalar quantizers, due to the subtraction of the dither upon decoding in Step D2 [45, Thms. 1-2].

We next bound the distance between the desired model  $\mathbf{w}_{t+\tau}^{\text{des}}$ , which is given by  $\mathbf{w}_{t+\tau}^{\text{des}} = \sum_{k=1}^K \alpha_k \tilde{\mathbf{w}}_{t+\tau}^{(k)}$  in (1.8), and the recovered one  $\mathbf{w}_{t+\tau}$ , as stated in the following theorem:

**THEOREM 1.3** (Thm. 2 of [53]) *When AS1 holds, the mean-squared distance between  $\mathbf{w}_{t+\tau}$  and  $\mathbf{w}_{t+\tau}^{\text{des}}$  satisfies*

$$\mathbb{E} \left\{ \|\mathbf{w}_{t+\tau} - \mathbf{w}_{t+\tau}^{\text{des}}\|^2 \right\} \leq M \zeta^2 \bar{\sigma}_{\mathcal{L}}^2 \tau \left( \sum_{t'=t}^{t+\tau-1} \eta_{t'}^2 \right) \sum_{k=1}^K \alpha_k^2 \xi_k^2. \quad (1.14)$$

Theorem 1.3 implies that the recovered model can be made arbitrarily close to the desired one by increasing  $K$ , namely, the number of users. For example, when  $\alpha_k = 1/K$ , i.e., conventional averaging, it follows from Theorem 1.3 that the mean-squared error in the weights decreases as  $1/K$ . In particular, if  $\max_k \alpha_k$  decreases with  $K$ , which essentially means that the updated model is not based only on a small part of the participating users, then the distortion vanishes in the aggregation process. Furthermore, when the step size  $\eta_t$  gradually decreases, which is known to contribute to the convergence of FL [36], it follows from Theorem 1.3 that the distortion decreases accordingly, further mitigating its effect as the FL iterations progress.

### Convergence Analysis

We next study the convergence of FL with UVeQFed. We do not restrict the labeled data of each of the users to be generated from an identical distribution, i.e., we consider a statistically heterogeneous scenario, thus faithfully representing FL setups [4, 54]. Such heterogeneity is in line with requirement A2, which does not impose any specific distribution structure on the underlying statistics of the training data. Following [36], we define the heterogeneity gap as

$$\psi \triangleq F(\mathbf{w}^o) - \sum_{k=1}^K \alpha_k \min_{\mathbf{w}} F_k(\mathbf{w}). \quad (1.15)$$

The value of  $\psi$  quantifies the degree of heterogeneity. If the training data originates from the same distribution, then  $\psi$  tends to zero as the training size grows. However, for heterogeneous data, its value is positive. The convergence of UVeQFed with federated averaging is characterized in the following theorem:

**THEOREM 1.4** (Thm. 3 of [53]) *Set  $\gamma = \tau \max(1, 4\rho_s/\rho_c)$  and consider a*

UVeQFed setup satisfying AS1-AS3. Under this setting, local SGD with step size  $\eta_t = \frac{\tau}{\rho_c(t+\gamma)}$  for each  $t \in \mathcal{N}$  satisfies

$$\mathbb{E}\{F(\mathbf{w}_t)\} - F(\mathbf{w}^\circ) \leq \frac{\rho_s}{2(t+\gamma)} \max\left(\frac{\rho_c^2 + \tau^2 b}{\tau \rho_c}, \gamma \|\mathbf{w}_0 - \mathbf{w}^\circ\|^2\right), \quad (1.16)$$

where

$$b \triangleq (1 + 4M\zeta^2\sigma_{\mathcal{L}}^2\tau^2) \sum_{k=1}^K \alpha_k^2 \xi_k^2 + 6\rho_s\psi + 8(\tau-1)^2 \sum_{k=1}^K \alpha_k \xi_k^2.$$

Theorem 1.4 implies that UVeQFed with local SGD, i.e., conventional federated averaging, converges at a rate of  $\mathcal{O}(1/t)$ . This is the same order of convergence as FL without quantization constraints for i.i.d. [35] as well as heterogeneous data [36, 55]. A similar order of convergence was also reported for previous probabilistic quantization schemes which typically considered i.i.d. data, e.g., [28, Thm. 3.2]. While it is difficult to identify the convergence gains of UVeQFed over previously proposed FL quantizers by comparing Theorem 1.4 to their corresponding convergence bounds, in the following section we empirically demonstrate that UVeQFed converges to more accurate global models compared to FL with probabilistic scalar quantizers, when trained using i.i.d. as well as heterogeneous data sets.

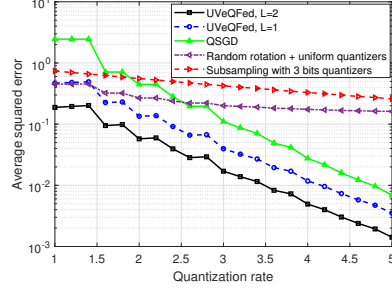
#### 1.4.2 Numerical Study

In this section we numerically evaluate UVeQFed. We first compare the quantization error induced by UVeQFed to competing methods utilized in FL. Then, we numerically demonstrate how the reduced distortion is translated in FL performance gains using both MNIST and CIFAR-10 data sets.

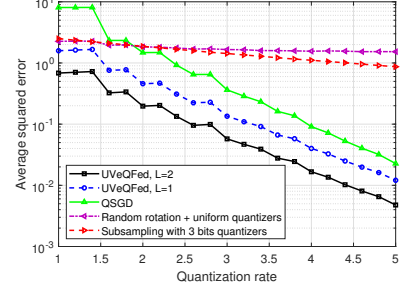
##### Quantization Distortion

We begin by focusing only on the compression method, studying its accuracy using synthetic data. We evaluate the distortion induced in quantization of UVeQFed operating with a two-dimensional hexagonal lattice, i.e.,  $L = 2$  and  $\mathbf{G} = [2, 0; 1, 1/\sqrt{3}]$  [56], as well as with scalar quantizers, namely,  $L = 1$  and  $\mathbf{G} = 1$ . The normalization coefficient is set to  $\zeta = \frac{2+\tilde{R}/5}{\sqrt{M}}$ , where  $\tilde{R}$  is the quantization rate, i.e., the ratio of the number of bits to the number of model updates entries  $m$ . The distortion of UVeQFed is compared to QSGD [28], which can be treated as UVeQFed without dither subtraction in Step D2 and with scalar quantizers, i.e.,  $L = 1$ . In addition, we evaluate the corresponding distortion achieved when using uniform quantizers with random unitary rotation [6], and to subsampling by random masks followed by uniform three-bit quantizers [6]. All the simulated quantization methods operate with the same quantization rate, i.e., the same overall number of bits.

Let  $\mathbf{H}$  be a  $128 \times 128$  matrix with Gaussian i.i.d. entries, and let  $\mathbf{\Sigma}$  be a  $128 \times 128$  matrix whose entries are given by  $(\mathbf{\Sigma})_{i,j} = e^{-0.2|i-j|}$ , representing



**Figure 1.9** Quantization distortion, i.i.d. data.



**Figure 1.10** Quantization distortion, correlated data.

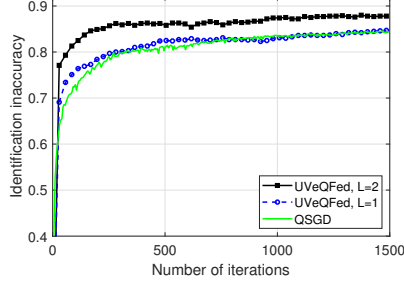
an exponentially decaying correlation. In Figs. 1.9-1.10 we depict the per-entry squared-error in quantizing  $\mathbf{H}$  and  $\Sigma\mathbf{H}\Sigma^T$ , representing independent and correlated data, respectively, versus the quantization rate  $\tilde{R}$ . The distortion is averaged over 100 independent realizations of  $\mathbf{H}$ . To meet the bit rate constraint when using lattice quantizers, we scaled  $\mathbf{G}$  such that the resulting codewords use less than  $128^2 \tilde{R}$  bits. For the scalar quantizers and subsampling-based scheme, the rate determines the quantization resolution and the subsampling ratio, respectively.

We observe in Figs. 1.9-1.10 that implementing the complete encoding-decoding steps E1-D4 allows UVeQFed to achieve a more accurate digital representation compared to previously proposed methods. It is also observed that UVeQFed with vector quantization outperforms its scalar counterpart, and that the gain is more notable when the quantized entries are correlated. This demonstrates the improved accuracy of jointly encoding multiple samples via vector quantization as well as the ability of UVeQFed to exploit statistical correlation in a universal manner by using fixed lattice-based quantization regions that do not depend on the underlying distribution.

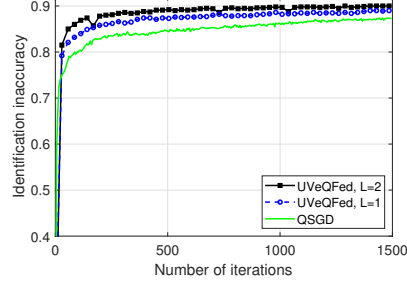
### Convergence Performance

Next, we demonstrate that the reduced distortion which follows from the combination of subtractive dithering and vector quantization in UVeQFed also translates into FL performance gains. To that aim, we evaluate UVeQFed for training neural networks using the MNIST and CIFAR-10 data sets, and compare its performance to that achievable using QSGD.

For MNIST, we use a fully-connected network with a single hidden layer of 50 neurons and an intermediate sigmoid activation. Each of the  $K = 15$  users has 1000 training samples, which are distributed sequentially among the users, i.e., the first user has the first 1000 samples in the data set, and so on, resulting in an uneven heterogeneous division of the labels of the users. The users update their weights using gradient descent, where federated averaging is carried out on each



**Figure 1.11** Convergence profile, MNIST,  $\tilde{R} = 2$ .



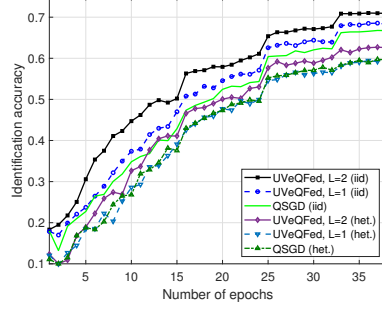
**Figure 1.12** Convergence profile, MNIST,  $\tilde{R} = 4$ .

iteration. The resulting accuracy versus the number of iterations is depicted in Figs. 1.11-1.12 for quantization rates  $\tilde{R} = 2$  and  $\tilde{R} = 4$ , respectively.

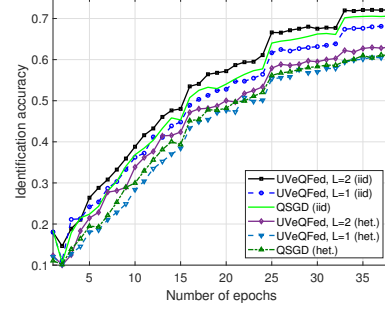
For CIFAR-10, we train the deep convolutional neural network architecture used in [57], whose trainable parameters constitute three convolution layers and two fully-connected layers. Here, we consider two methods for distributing the 50000 training images of CIFAR-10 among the  $K = 10$  users: An i.i.d. division, where each user has the same number of samples from each of the 10 labels, and a heterogeneous division, in which at least 25% of the samples of each user correspond to a single distinct label. Each user completes a single epoch of SGD with mini-batch size 60 before the models are aggregated. The resulting accuracy versus the number of epochs is depicted in Figs. 1.13-1.14 for quantization rates  $\tilde{R} = 2$  and  $\tilde{R} = 4$ , respectively.

We observe in Figs. 1.11-1.14 that UVeQFed with vector quantizer, i.e.,  $L = 2$ , results in convergence to the most accurate model for all the considered scenarios. The gains are more dominant for  $\tilde{R} = 2$ , implying that the usage of UVeQFed with multi-dimensional lattices can notably improve the performance over low rate channels. Particularly, we observe in Figs. 1.13-1.14 that similar gains of UVeQFed are noted for both i.i.d. as well as heterogeneous setups, while the heterogeneous division of the data degrades the accuracy of all considered schemes compared to the i.i.d. division. It is also observed that UVeQFed with scalar quantizers, i.e.,  $L = 1$ , achieves improved convergence compared to QSGD for most considered setups, which stems from its reduced distortion.

The results presented in this section demonstrate that the theoretical benefits of UVeQFed, which rigorously hold under AS1-AS3, translate into improved convergence when operating under rate constraints with non-synthetic data. They also demonstrate how introducing the state-of-the-art quantization theoretic concepts of subtractive dithering and universal joint vector quantization contributes to both reducing the quantization distortion as well as improving FL convergence.



**Figure 1.13** Convergence profile, CIFAR-10,  $\tilde{R} = 2$ .



**Figure 1.14** Convergence profile, CIFAR-10,  $\tilde{R} = 4$ .

## 1.5 Summary

In the emerging machine learning paradigm of FL, the task of training highly-parameterized models becomes the joint effort of multiple remote users, orchestrated by the server in a centralized manner. This distributed learning operation, which brings forth various gains in terms of privacy, also gives rise to a multitude of challenges. One of these challenges stems from the reliance of FL on the ability of the participating entities to reliably and repeatedly communicate over channels whose throughput is typically constrained, motivating the users to convey their model updates to the server in a compressed manner with minimal distortion.

In this chapter we have considered the problem of model update compression from a quantization theory perspective. We first identified the unique characteristics of FL in light of which quantization methods for FL should be designed. Then, we presented established concepts in quantization theory which facilitate uplink compression in FL. These include *a*) the usage of probabilistic (dithered) quantization to allow the distortion to be reduced in federated averaging; *b*) the integration of subtractive dithering to reduce the distortion by exploiting a random seed shared by the server and each user; *c*) the extension to vector quantizers in a universal manner to further improve the rate-distortion tradeoff; and *d*) the combination of lossy quantization with lossless source coding to exploit non-uniformity and sparsity of the digital representations. These concepts are summarized in a concrete and systematic FL quantization scheme, referred to as UVeQFed. We analyzed UVeQFed, proving that its error term is mitigated by federated averaging. We also characterized its convergence profile, showing that its asymptotic decay rate is the same as an unquantized local SGD. Our numerical study demonstrates that UVeQFed achieves more accurate recovery of model updates in each FL iteration compared to previously proposed schemes for the same number of bits, and that its reduced distortion is translated into improved convergence with the MNIST and CIFAR-10 data sets.

## References

- [1] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, p. 436, 2015.
- [2] J. Chen and X. Ran, “Deep learning with edge computing: A review,” *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1655–1674, 2019.
- [3] H. B. McMahan, E. Moore, D. Ramage, and S. Hampson, “Communication-efficient learning of deep networks from decentralized data,” *arXiv preprint arXiv:1602.05629*, 2016.
- [4] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, “Advances and open problems in federated learning,” *arXiv preprint arXiv:1912.04977*, 2019.
- [5] J. Konečný, H. B. McMahan, D. Ramage, and P. Richtárik, “Federated optimization: Distributed machine learning for on-device intelligence,” *arXiv preprint arXiv:1610.02527*, Oct. 2016.
- [6] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, “Federated learning: Strategies for improving communication efficiency,” *arXiv preprint arXiv:1610.05492*, 2016.
- [7] M. Chen, H. V. Poor, W. Saad, and S. Cui, “Wireless communications for collaborative federated learning in the internet of things,” *IEEE Communications Magazine*, to appear, 2020.
- [8] N. H. Tran, W. Bao, A. Zomaya, N. Minh N.H., and C. S. Hong, “Federated learning over wireless networks: Optimization model design and analysis,” in *Proc. IEEE Conference on Computer Communications*, Paris, France, April 2019.
- [9] M. Chen, H. V. Poor, W. Saad, and S. Cui, “Convergence time optimization for federated learning over wireless networks,” *arXiv preprint arXiv:2001.07845*, 2020.
- [10] T. T. Vu, D. T. Ngo, N. H. Tran, H. Q. Ngo, M. N. Dao, and R. H. Middleton, “Cell-free massive mimo for wireless federated learning,” *IEEE Transactions on Wireless Communications*, to appear, 2020.
- [11] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, “Energy efficient federated learning over wireless communication networks,” *arXiv preprint arXiv:1911.02417*, 2019.
- [12] G. Zhu, D. Liu, Y. Du, C. You, J. Zhang, and K. Huang, “Toward an intelligent edge: Wireless communication meets machine learning,” *IEEE Communications Magazine*, vol. 58, no. 1, pp. 19–25, Jan. 2020.
- [13] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, “Adaptive federated learning in resource constrained edge computing systems,”

- 
- IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, June 2019.
- [14] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, “A joint learning and communications framework for federated learning over wireless networks,” *IEEE Transactions on Wireless Communications*, to appear, 2020.
  - [15] J. Ren, G. Yu, and G. Ding, “Accelerating DNN training in wireless federated edge learning system,” *arXiv preprint arXiv:1905.09712*, 2019.
  - [16] Q. Zeng, Y. Du, K. Huang, and K. K. Leung, “Energy-efficient resource management for federated edge learning with CPU-GPU heterogeneous computing,” *arXiv preprint arXiv:2007.07122*, 2020.
  - [17] R. Jin, X. He, and H. Dai, “On the design of communication efficient federated learning over wireless networks,” *arXiv preprint arXiv:2004.07351*, 2020.
  - [18] M. M. Amiri and D. Gunduz, “Machine learning at the wireless edge: Distributed stochastic gradient descent over-the-air,” in *Proc. of IEEE International Symposium on Information Theory (ISIT)*, Paris, France, July 2019.
  - [19] G. Zhu, Y. Du, D. Gunduz, and K. Huang, “One-bit over-the-air aggregation for communication-efficient federated edge learning: Design and convergence analysis,” *arXiv preprint arXiv:2001.05713*, 2020.
  - [20] T. Sery, N. Shlezinger, K. Cohen, and Y. C. Eldar, “COTAF: Convergent over-the-air federated learning,” in *Proc. IEEE Global Communications Conference (GLOBECOM)*, 2020.
  - [21] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, “Deep gradient compression: Reducing the communication bandwidth for distributed training,” *arXiv preprint arXiv:1712.01887*, 2017.
  - [22] C. Hardy, E. Le Merrer, and B. Sericola, “Distributed deep learning on edge-devices: feasibility via adaptive compression,” in *Proc. IEEE International Symposium on Network Computing and Applications (NCA)*, 2017.
  - [23] A. F. Aji and K. Heafeld, “Sparse communication for distributed gradient descent,” *arXiv preprint arXiv:1704.05021*, 2017.
  - [24] D. Alistarh, T. Hoeffler, M. Johansson, N. Konstantinov, S. Khirirat, and C. Renggli, “The convergence of sparsified gradient methods,” in *Neural Information Processing Systems*, 2018, pp. 5973–5983.
  - [25] P. Han, S. Wang, and K. K. Leung, “Adaptive gradient sparsification for efficient federated learning: An online learning approach,” *arXiv preprint arXiv:2001.04756*, 2020.
  - [26] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar, “SignSGD: Compressed optimisation for non-convex problems,” *arXiv preprint arXiv:1802.04434*, 2018.
  - [27] 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 *Neural Information Processing Systems*, 2017, pp. 1509–1519.
  - [28] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, “QSGD: Communication-efficient SGD via gradient quantization and encoding,” in *Neural Information Processing Systems*, 2017, pp. 1709–1720.
  - [29] S. Horvath, C.-Y. Ho, L. Horvath, A. N. Sahu, M. Canini, and P. Richtarik, “Natural compression for distributed deep learning,” *arXiv preprint arXiv:1905.10988*, 2019.



- 
- [30] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, “Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization,” *arXiv preprint arXiv:1909.13014*, 2019.
  - [31] S. P. Karimireddy, Q. Rebjock, S. U. Stich, and M. Jaggi, “Error feedback fixes signSGD and other gradient compression schemes,” in *International Conference on Machine Learning*, 2019, pp. 3252–3261.
  - [32] R. M. Gray and D. L. Neuhoff, “Quantization,” *IEEE Transactions on Information Theory*, vol. 44, no. 6, pp. 2325–2383, 1998.
  - [33] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. John Wiley & Sons, 2012.
  - [34] Y. Polyanskiy and Y. Wu, “Lecture notes on information theory,” *Lecture Notes for 6.441 (MIT), ECE563 (University of Illinois Urbana-Champaign), and STAT 664 (Yale)*, 2012–2017.
  - [35] S. U. Stich, “Local SGD converges fast and communicates little,” *arXiv preprint arXiv:1805.09767*, 2018.
  - [36] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, “On the convergence of fedavg on non-iid data,” *arXiv preprint arXiv:1907.02189*, 2019.
  - [37] B. Woodworth, K. K. Patel, S. U. Stich, Z. Dai, B. Bullins, H. B. McMahan, O. Shamir, and N. Srebro, “Is local SGD better than minibatch SGD?” *arXiv preprint arXiv:2002.07839*, 2020.
  - [38] J. Zhang, C. De Sa, I. Mitliagkas, and C. Ré, “Parallel SGD: When does averaging help?” *arXiv preprint arXiv:1606.07365*, 2016.
  - [39] L. Zhu, Z. Liu, and S. Han, “Deep leakage from gradients,” in *Advances in Neural Information Processing Systems*, 2019, pp. 14 774–14 784.
  - [40] speedtest.net, “Speedtest united states market report,” 2019. [Online]. Available: <http://www.speedtest.net/reports/united-states/>
  - [41] S. Horváth, D. Kovalev, K. Mishchenko, S. Stich, and P. Richtárik, “Stochastic distributed learning with gradient quantization and variance reduction,” *arXiv preprint arXiv:1904.05115*, 2019.
  - [42] A. El Gamal and Y.-H. Kim, *Network Information Theory*. Cambridge University Press, 2011.
  - [43] A. Wyner and J. Ziv, “The rate-distortion function for source coding with side information at the decoder,” *IEEE Transactions on Information Theory*, vol. 22, no. 1, pp. 1–10, 1976.
  - [44] S. P. Lipshitz, R. A. Wannamaker, and J. Vanderkooy, “Quantization and dither: A theoretical survey,” *Journal of the Audio Engineering Society*, vol. 40, no. 5, pp. 355–375, 1992.
  - [45] R. M. Gray and T. G. Stockham, “Dithered quantizers,” *IEEE Transactions on Information Theory*, vol. 39, no. 3, pp. 805–812, 1993.
  - [46] B. Widrow, I. Kollar, and M.-C. Liu, “Statistical theory of quantization,” *IEEE Transactions on Instrumentation and Measurement*, vol. 45, no. 2, pp. 353–361, 1996.
  - [47] J. Ziv, “On universal quantization,” *IEEE Transactions on Information Theory*, vol. 31, no. 3, pp. 344–347, 1985.
  - [48] R. Zamir and M. Feder, “On universal quantization by randomized uniform/lattice quantizers,” *IEEE Transactions on Information Theory*, vol. 38, no. 2, pp. 428–436, 1992.

- [49] —, “On lattice quantization noise,” *IEEE Transactions on Information Theory*, vol. 42, no. 4, pp. 1152–1159, 1996.
- [50] J. H. Conway and N. J. A. Sloane, *Sphere Packings, Lattices and Groups*. Springer Science & Business Media, 2013, vol. 290.
- [51] Y. Zhang, J. C. Duchi, and M. J. Wainwright, “Communication-efficient algorithms for statistical optimization,” *The Journal of Machine Learning Research*, vol. 14, no. 1, pp. 3321–3363, 2013.
- [52] J. Conway and N. Sloane, “Voronoi regions of lattices, second moments of polytopes, and quantization,” *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 211–226, 1982.
- [53] N. Shlezinger, M. Chen, Y. C. Eldar, H. V. Poor, and S. Cui, “UVeQFed: Universal vector quantization for federated learning,” *IEEE Transactions on Signal Processing*, early access, 2020.
- [54] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, “Federated learning: Challenges, methods, and future directions,” *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, May 2020.
- [55] A. Koloskova, N. Loizou, S. Boreiri, M. Jaggi, and S. U. Stich, “A unified theory of decentralized SGD with changing topology and local updates,” *arXiv preprint arXiv:2003.10422*, 2020.
- [56] A. Kirac and P. Vaidyanathan, “Results on lattice vector quantization with dithering,” *IEEE Transactions on Circuits and Systems—Part II: Analog and Digital Signal Processing*, vol. 43, no. 12, pp. 811–826, 1996.
- [57] MathWorks Deep Learning Toolbox Team, “Deep learning tutorial series,” *MATLAB Central File Exchange*, 2020. [Online]. Available: <https://www.mathworks.com/matlabcentral/fileexchange/62990-deep-learning-tutorial-series>