This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2024.3488377

Approximate Wireless Communication for Lossy Gradient Updates in IoT Federated Learning

Xiang Ma, *Student Member, IEEE,* Haijian Sun, *Senior Member, IEEE,* Rose Qingyang Hu, *Fellow, IEEE,* and Yi Qian, *Fellow, IEEE*

***Abstract*—Federated learning (FL) has emerged as a distributed machine learning (ML) technique that can protect local data privacy for participating clients and improve system efficiency. Instead of sharing raw data, FL exchanges intermediate learning parameters, such as gradients, among clients. This article presents an efficient wireless communication approach tailored for FL parameter transmission, especially for Internet of Things (IoT) devices, to facilitate model aggregation. Our study considers practical wireless channels that can lead to random bit errors, substantially affecting FL performance. Motivated by empirical gradient value distribution, we introduce a novel received bit masking method that confines received gradient values within prescribed limits. Moreover, given the intrinsic error resilience of ML gradients, our approach enables the delivery of approximate gradient values with errors without resorting to extensive error correction coding or retransmission. This strategy reduces computational overhead at both the transmitter and the receiver and minimizes communication latency. Consequently, our scheme is particularly well-suited for resource-constrained IoT devices. Our simulations demonstrate that our proposed scheme can effectively mitigate random bit errors in FL performance, achieving similar learning objectives but with the 50% air time required by existing methods involving error correction and retransmission.**

***Index Terms*—Approximate communication, federated learning, lossy wireless communication, gradient model updates, forward error correction (FEC).**

I. Introduction

Federated learning (FL) [1] is a promising learning paradigm that allows clients to perform machine learning (ML) tasks locally while benefiting from the collective learning capabilities of other clients through the exchange of model parameters. Specifically, client data remains local, and only the learned model is shared, ensuring robust privacy protection for all participants. The FL system comprises a central parameter server (PS) and multiple clients. The server aggregates the local models uploaded by the clients and sends the global model back.

Manuscript received April xx, 2024; revised June xx, 2024. This article was partly presented at the ACM Workshop on Wireless Security and Machine Learning (WiseML 2023).

Xiang Ma and Rose Qingyang Hu are with the Department of Electrical and Computer Engineering, Utah State University, Logan, UT 84322 USA (e-mail: xiang.ma@ieee.org, rose.hu@usu.edu).

Haijian Sun is with the School of Electrical and Computer Engineering, University of Georgia, Athens, GA 30602 USA (e-mail: hsun@uga.edu).

Yi Qian is with the Department of Electrical and Computer Engineering, University of Nebraska–Lincoln, Omaha, NE 68182 USA (e-mail: yi.qian@unl.edu).

In today's computation world, a paradigm shift is the transition from cloud to edge computation. Due to computing scaling law, edge devices, even mobile IoTs, are capable of processing fairly complex learning and inference tasks. The topic in this paper is to consider a distributed learning framework, such as FL, collaboratively accomplished among IoT devices [2]. In such a setting, FL model exchange, i.e., the intermediate model results will be sent wirelessly to the aggregator (FL server) during each training round. However, the nature of random wireless channels can introduce errors during data transmission. The erroneous transmission could result in meaningless learning. A study by the authors in [3] focused on transmission bit errors in FL, specifically in a packet erasure channel. It shows the mean square error of the ground truth model and the error model experiences large fluctuation. And the error model does not converge. Numerous methods have been proposed to address this issue. For example, in the physical layer, high-power transmission can enhance signal quality and overcome noise effects. The receiver equalization can mitigate the impact of fading channels. In the upper layers, error detection and correction methods such as parity check, checksum check, cyclic redundancy check (CRC), and forward error correction (FEC) [4] are commonly utilized. FEC encodes the message with redundant bits as an error correction code (ECC) to correct certain errors on the receiver side. Common ECCs include block code, convolutional code, and low-density parity check (LDPC) code. However, it introduces redundancy and requires additional computations on the transmitter and receiver for encoding and decoding the message [5]. Furthermore, communication overhead is introduced to transmit redundancy information. When the channel conditions are poor, errors may exceed the ECC's correction capability. In such cases, packet retransmission is employed in the transport layer to ensure reliable transmission. FEC and packet retransmission can intensely drain energy-constrained IoT devices [6]. Also, the computation and communication resources on IoT devices are constrained. Extra methods are needed to reduce the communication costs of FL in wireless networks.

Numerous schemes have been devised in FL to mitigate communication overhead. To allow large-step model updates to be uploaded at once, Federated Averaging (FedAvg) [7] groups multiple stochastic gradient descent (SGD) updates together. However, transmitting gradients can still lead to significant delays for large-scale distributed ML models with millions of parameters. In [8] and [9], a joint problem of

Authorized licensed use limited to: Utah State University. Downloaded on December 26,2024 at 18:25:55 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2024.3488377

resource allocation and user selection is proposed to minimize the convergence time in FL. Advanced transmission schemes such as non-orthogonal multiple access (NOMA) [10] offer promising solutions. NOMA allows various users in the FL system to transmit model parameters simultaneously, thus improving channel efficiency and reducing communication delay. Another practical approach to address this challenge is gradient compression, with extensive research demonstrating the efficacy of gradient sparsification and quantization while incurring minimal performance loss. 1-bit SGD was applied in [11] to reduce the transmission size of the gradient in a distributed data-parallel system, achieving at least $4\times$ reduction in the real training time from end to end. Furthermore, in [12], the authors found that the drop in 99% of the gradients results in no or negligible loss of precision in multiple data sets. A joint learning and communication problem is formulated in [13] to improve the energy efficiency for IoT devices.

Approximate communication, an analogy of approximate computing techniques in parallel systems, reduces communication overhead between transmitters and receivers [14]. This approach permits communications to be executed "approximately", deliberately allowing minor errors to achieve efficiency gains. However, the system must possess intrinsic error resiliency, and the error should be acceptable to adopt approximate communication successfully. Within the FL system, error resiliency manifests itself in various aspects. First, the gradients in ML exhibit minimal variation, as demonstrated by empirical studies in [15]–[17]. These studies reveal that gradient values often fall within the $(-1, 1)$ or even narrower, such as $(-0.01, 0.01)$. [15] provides histograms of gradients for fully connected and convolutional layers in different training iterations, while [16] presents probability density functions (PDFs) and cumulative distribution functions (CDFs) of gradients. These sources collectively establish that the ML gradients are distributed within a small range and that this range is empirically known. Furthermore, the error resilience of the FL system is demonstrated through gradient compression, which introduces errors in the form of gradient quantization. Individual gradient errors arise during quantization. However, while individual gradients are accurately represented, a fraction of all gradients are removed through gradient sparsification. Finally, the FL system's model aggregation occurs on average, enhancing error resilience by limiting errors to an acceptable level. With more clients participating in the learning process, the resilience to errors improves.

Approximate communication has been applied in various domains, such as network-on-chip (NoC) designs, contributing significantly to higher energy efficiency [18]–[20]. In [18], an approximate communication framework was designed for photonic NoCs to reduce the overhead of lasers and turning power while maintaining acceptable distortion levels. Furthermore, [19] introduced a quality control method that collaborates with data approximation mechanisms to reduce packet size, thus reducing network power consumption and latency. [20] proposed packet production and reduction in error checking for NoC as approximate communication solutions. The proposed scheme achieves better performance in both energy consumption and latency. Approximate communication was also presented in [21] to eliminate the need for additional network or spectrum resources for redundant information transmission in wireless media delivery. By prioritizing significant bits in wireless packets and placing them in the most significant bit positions during high-order modulation, the proposed method significantly improved video quality by 5 to 20 dB under wireless conditions. Moreover, in [22], researchers presented a distributed approximate Newton-type method to improve communication efficiency in stochastic optimization and learning problems, accelerating the convergence speed of quadratic problems.

Inspired by these works, we introduce an approximate wireless communication framework for FL. This framework encourages the lossy transmission of FL model gradients, offering the advantage of low latency, reduced overhead, and reduced computation. In [3], the server discards erroneous local models in this scenario and resorts to past local models for continuity. This will cause the most recent model updates to be lost. In [23], the authors proposed the FedLC framework, which applied the user datagram protocol (UDP) rather than the transmission control protocol (TCP) as the basis for transmitting the model update in a lossy communication channel. FEC and packet retransmission are also used as countermeasures to prevent packet losses. Unlike [23], our proposed approximate wireless communication operates in the physical layer, while FedLC uses FEC in the application layer and packet retransmission, UDP in the transport layer. They can work together to achieve better performance.

In this research, we conducted a comprehensive theoretical analysis of ML gradients. Specifically, we analyze the gradients in fully connected neural networks and convolutional neural networks (CNN) under commonly employed structures like Sigmoid/ReLU activation and cross-entropy loss functions. This enables us to clarify the gradient vanishing and gradient exploding problems in deep neural networks (DNN). The main contributions of this study are summarized below.

1) This study offers a sketch mathematical analysis of gradient values in commonly employed ML settings. We demonstrate that back-propagation could result in gradient vanishing and gradient exploding problems.
2) Using prior knowledge of gradients and the effectiveness of gradient clipping, received bits are thoughtfully masked to confine errors within a small range. This allows approximate transmission for error-resilient FL systems.
3) Approximate communication is applied in wireless FL gradient transmission, effectively reducing communication latency and computational overhead. Additionally, using gray coding with high-order modulation offers protection for the most significant bits (MSBs), resulting in notable improvements in learning performance. Gradient sparsification can further reduce communication costs and mitigate the side effects caused by approximate communication.
4) Extensive simulations are conducted to validate the efficacy of the proposed approximate wireless communication for FL gradients. Learning performance is evaluated by considering factors such as the number of users, the size of the error, and the modulation index that can

Authorized licensed use limited to: Utah State University. Downloaded on December 26,2024 at 18:25:55 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2024.3488377

influence the results.

The rest of the paper is organized as follows. Section II provides an introduction to the system model, including both the FL system model and the wireless channel model. Section III presents the mathematical analysis of gradients in commonly used ML settings. Section IV describes the proposed approximate transmission method. Section V presents the simulation results, and finally, Section VI concludes the paper.

II. System Model

In this section, we begin by introducing the FL system, followed by the wireless channel model. Specifically, we consider an FL system comprising M edge clients, each connected to the server wirelessly. The entire dataset D is distributed among these M clients, each client m containing a subset of data denoted as D_m, i.e., $\sum_m |D_m| = |D|$, where $|D_m|$ and $|D|$ represent the size of dataset D_m and D.

A. FL System

FL operates as an iterative ML algorithm between the server and the clients, progressing round by round. Each round entails four fundamental steps. The system model is illustrated in Fig. 1. Wireless communication is used for the downlink in step (S1) and the uplink in step (S3). In step (S2), the clients perform local learning. The global aggregation at the server happens in step (S4). The participating clients vary in computational capabilities, and the channel conditions between the clients and the server also differ.

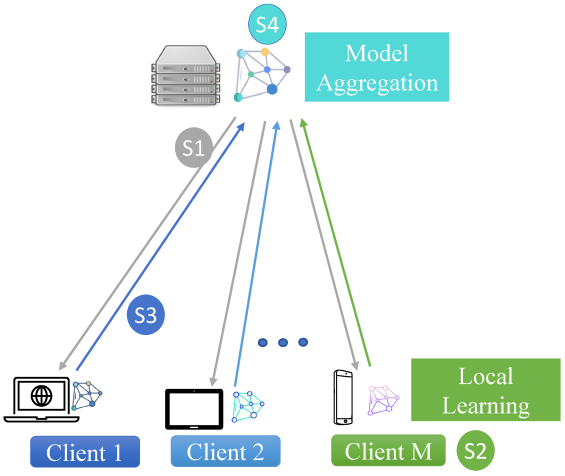


Fig. 1: FL system model

The objective function of FL can be defined as:

$$\min_{\boldsymbol{w}\in R^d} f(\boldsymbol{w}) \quad \text{where} \quad f(\boldsymbol{w}) \overset{\text{def}}{=} \frac{1}{|D|}\sum_{i=1}^{|D|} f_i(\boldsymbol{w}), \tag{1}$$

The function $f_i(\boldsymbol{w}) = C(\boldsymbol{x_i}, \boldsymbol{y_i}; \boldsymbol{w})$ serves as the cost or loss function, quantifying the inference error between the data sample $(\boldsymbol{x_i}, \boldsymbol{y_i})$ and the inference produced by model parameters $\boldsymbol{w}$. The commonly employed loss function for classification problems in ML is the cross-entropy function, particularly in neural network models. In multiclass classification, the label $\boldsymbol{y_i}$ is often one-hot encoded, ensuring that each label carries equal weight. Thus, $\boldsymbol{y_i} = [0, \ldots, 0, 1, 0, \ldots, 0]^T$, with the value 1 located at the i-th position.

When the data are distributed among M clients, the objective function (1) can be rewritten as follows:

$$f(\boldsymbol{w}) = \sum_{m=1}^{M} \frac{|D_m|}{|D|} F_m(\boldsymbol{w}), \tag{2}$$

where $F_m(\boldsymbol{w}) = \frac{1}{|D_m|}\sum_{i\in D_m} f_i(\boldsymbol{w})$, $\frac{|D_m|}{|D|}$ denotes the aggregation coefficient, ensuring that $\sum_m \frac{|D_m|}{|D|} = 1$. Distributing data and computation across multiple clients can convert a traditional centralized ML problem into a distributed FL problem.

Given that the cost function for neural networks is typically non-convex, directly solving for the global minimum poses challenges. Consequently, the gradient descent method is commonly used in ML to identify local minimum points. Stochastic gradient descent (SGD), a variant of the gradient descent method, is beneficial in avoiding local minimums by randomly selecting data samples. As a result, gradients play a central and crucial role in the learning process. In each round, the local gradient at each client can be expressed as follows:

$$g_t^m = \nabla F_m(w_t). \tag{3}$$

g_t^m is the local gradient of the client m in the round t. The global gradient after aggregation in round t is

$$g_t = \sum_{m=1}^{M} \frac{|D_m|}{|D|} g_t^m. \tag{4}$$

The server maintains the model weights of the last round, denoted w_t, and updates the global model using the following procedure:

$$w_{t+1} = w_t - \eta g_t. \tag{5}$$

Here, η is the learning rate, typically within the $(0, 1)$ range.

B. Wireless Channel Model

In the context of FL, we consider the fading channel in the uplink, as illustrated in step (S3) in Fig. 1, which may lead to random bit errors. In contrast, for the downlink, we assume that the server can provide global gradients to clients with negligible errors, for example, with higher transmit power that leads to a higher signal-to-noise ratio (SNR) [3].

In the uplink, a time division scheme can be employed, where each user is assigned a specific time slot while sharing the same channel. The signal received on the server can be expressed as follows:

$$r_t^m = \sqrt{p_t^m (d^m)^{-\alpha}} h_t^m g_t^m + n_t^m, \tag{6}$$

where r_t^m denotes the signal received at the server from client m during round t. Transmission power is represented by p_t^m, and small-scale fading by h_t^m, assumed to follow a complex standard Gaussian distribution, that is, $h_t^m \sim \mathcal{CN}(0, 1)$. Large-scale fading is explained by path loss, the distance between the

Authorized licensed use limited to: Utah State University. Downloaded on December 26,2024 at 18:25:55 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2024.3488377

server and the client m denoted as d^m, and the path loss exponent represented by α. Additive noise is assumed to follow a Gaussian distribution, $n^t \sim \mathcal{CN}(0, n_\sigma^2)$, where n_σ^2 represents the variance. Additionally, we assume that the server has channel information, that is, $c_t^m = \sqrt{p_t^m (d^m)^{-\alpha}} h_t^m$, with only the noise remaining unknown.

The transmission process can be described below. First, the gradients are converted from decimal to binary format. Subsequently, the bits are mapped to symbols using a quadrature amplitude modulation (QAM) scheme. The symbols are then transmitted through the wireless fading channel. At the receiver side, the signal is decoded and demodulated to the closest point in the QAM constellation. The demodulation process can be represented as follows:

$$\hat{g}_t^m = \arg_{\bar{g}_t^m \in \mathcal{G}} \min ||r_t^m - \sqrt{p_t^m (d^m)^{-\alpha}} h_t^m \bar{g}_t^m||^2, \quad (7)$$

where $\mathcal{G}$ refers to the set of symbol points on the constellation diagram. $\bar{g}_t^m$ represents a potential symbol point within $\mathcal{G}$, while $\hat{g}_t^m$ denotes the optimal symbol point.

The main notation in the paper is summarized in Table I.

TABLE I: Summary of notations

Notation	Definition
M; m	The total number of clients connected to the server; the client index
L; l	The last layer of the neural network; the neural network layer index
$\boldsymbol{x}$; $\boldsymbol{y}$; $\hat{y}$	Features of a data point sample; corresponding true label of the data point; the predicted label
w; g; b	Model weight; model gradient; neural network bias
η; δ	Learning rate; intermediate quantity as "error"
C; $\sigma(\cdot)$	Loss function; activation function
z; a	Intermediate output of neuron; neuron output
N_l; N_g; N	The number of neurons in the l-th layer; the number of gradients in neural network; the number of data samples
i; j; k; p; q	Index
s; t; u, v	Index
t	Time (round) index
h; α; n; r	Channel factor; path loss exponent; noise; received signal
p_t^m; d^m	The transmission power of the client m at time t; Distance between the client m and the server

III. Gradients Analysis with Back-propagation

ML model training aims to find the optimal point to minimize loss functions. Since neural networks are usually non-convex, SGD and back-propagation are typical methods for model training. However, gradient vanishing or exploding problems can occur in deep neural networks, preventing efficient model learning. The transmission of gradients with errors can cause the gradient to change in random directions. The received gradients can be substantial, making the model unstable. Studying the gradient behavior and the existing method to prevent gradient vanishing/exploding helps to design the method to perform approximate communication.

A. Gradient in Fully Connected Neural Networks

A neuron in neural networks serves as the fundamental unit for data processing, taking inspiration from neurons in human brains. It accepts input from the preceding layers, processes them, and transmits them to the next layer [24]. The diagram depicting neurons is illustrated in Fig. 2, providing insight into the input and output of the j-th neuron in the l -th layer.

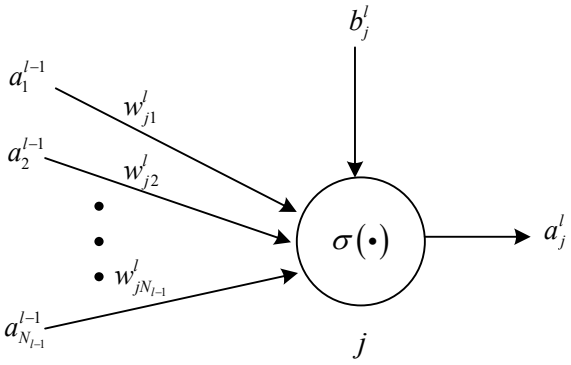


Fig. 2: Artificial neuron diagram

In the ML field, especially within neural networks, SGD and backpropagation are widely used optimization techniques for locating global minimum points. In a fully connected neural network, the feedforward equation at each neuron can be expressed as follows:

$$\begin{aligned} z_j^l &= b_j^l + \sum_k^{N_{l-1}} w_{jk}^l a_k^{l-1}, \\ a_j^l &= \sigma(z_j^l). \end{aligned} \quad (8)$$

Here, b represents the bias, w denotes the weights, z stands for the intermediate output, and a means the neuron output after passing through the activation function $\sigma(\cdot)$. The indices j and l denote that this neuron is the j-th neuron located in the l-th layer of the neural network. Additionally, k is the index of the input from the $(l-1)$-th layer, and N_{l-1} is the number of neurons in the $(l-1)$-th layer. In particular, the neuron output from the previous layers serves as the input to the current layer, and this process can continue back to the first layer, as illustrated in Fig. 2. And equation (8) becomes

$$\begin{aligned} z_j^1 &= b_j^1 + \sum_k^{N_0} w_{jk}^1 x_k^0, \\ a_j^1 &= \sigma(z_j^1). \end{aligned} \quad (9)$$

in the first layer. x_k^0 is the k-th input, and the superscript 0 is used to maintain writing consistency. The corresponding four fundamental equations in backpropagation for a fully connected neural network are [25]

$$\delta_j^L = \frac{\partial C}{\partial z_j^L} = \frac{\partial C}{\partial a_j^L} \frac{\partial a_j^L}{\partial z_j^L} = \frac{\partial C}{\partial a_j^L} \sigma'(z_j^L), \quad (10a)$$

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} = \sum_k^{N_{l+1}} \frac{\partial C}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial a_j^l} \frac{\partial a_j^l}{\partial z_j^l} = \sum_k^{N_{l+1}} \delta_k^{l+1} w_{kj}^{l+1} \sigma'(z_j^l), \quad (10b)$$

$$\frac{\partial C}{\partial b_j^l} = \frac{\partial C}{\partial z_j^l} \frac{\partial z_j^l}{\partial b_j^l} = \delta_j^l, \quad (10c)$$

$$\frac{\partial C}{\partial w_{jk}^l} = \frac{\partial C}{\partial z_j^l} \frac{\partial z_j^l}{\partial w_{jk}^l} = \delta_j^l a_k^{l-1}. \quad (10d)$$

Authorized licensed use limited to: Utah State University. Downloaded on December 26,2024 at 18:25:55 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2024.3488377

Here, L designates the final layer in the neural network, and δ_j^l signifies the "error" in the l-th layer in the neuron j. δ_j^l serves as an intermediate quantity introduced essential for calculating the partial derivatives in equations (10c) and (10d).

The weight update can be rewritten as

$$w_{jk}^l = w_{jk}^l - \eta \delta_j^l a_k^{l-1}. \tag{11}$$

Similarly, the bias update can be written as

$$b_j^l = b_j^l - \eta \delta_j^l. \tag{12}$$

The calculation of the gradient $g^l = \nabla_{\boldsymbol{w}^l} C(\boldsymbol{x_i}, \boldsymbol{y_i}; \boldsymbol{w^l}) = [\frac{\partial C}{\partial w_{j_1 k_1}^l}, \frac{\partial C}{\partial w_{j_1 k_2}^l}, \ldots, \frac{\partial C}{\partial w_{N_l N_{l-1}}^l}]$ in equation (10d) involves two terms, δ_j^l and a_k^{l-1}. These two terms are discussed separately. Firstly, a_k^{l-1} denotes the output of the activation function of the neuron k in the $(l-1)$-th layer. Its range depends on the specific activation function used. For example, the Sigmoid function ensures that a_k^{l-1} falls within the range $(0, 1)$ regardless of input z_k^{l-1}. While for the ReLU activation function, the output of the activation function a_k^{l-1} depends on the input z_k^{l-1}. More detailed analysis and mathematical expressions on activation functions can be found in [26].

The calculation of the other term δ_j^l is described in equation (10b), which involves a sum of products between the errors of the next layer δ_k^{l+1}, the weights of the node j in the l -th layer to the node k in the subsequent layer w_{kj}^{l+1}, and the derivative of the activation function $\sigma'(z_j^l)$. And the summation counts through all neurons in the $(l+1)$-th layer. The following analysis will go through the third term $\sigma'(z_j^l)$ first, then w_{kj}^{l+1}, finally δ_k^{l+1}.

Firstly, the derivative of the activation function $\sigma'(z_j^l)$ depends on the specific activation function used, with values ranging from $(0, 0.25)$ for the Sigmoid function and $\{0, 1\}$ for ReLU. Next, the weight w_{kj}^{l+1} is based on the initialization of the model, the learning rate η, and the previous round gradients based on Equation (5). Common weight initialization methods generate random weight values within the range of $(-1, 1)$ [27], drawn from Gaussian or uniform distributions [28] in a heuristic manner. Furthermore, more recent initialization methods improve random initialization, such as in [29], where the weight is initialized following a uniform distribution $w \sim U[\frac{1}{\sqrt{N_p}}, \frac{1}{\sqrt{N_p}}]$, N_p is the number of neurons in the previous layer. The weight is in the $[\frac{1}{\sqrt{N_p}}, \frac{1}{\sqrt{N_p}}]$ range. And in [30], the weight is initialized in the Gaussian distribution, $w \sim \mathcal{N}(0, \sqrt{(2/N_p)})$, where 99.7% of the weights are within $[3 * \sqrt{(2/N_p)}, 3 * \sqrt{(2/N_p)}]$. Lastly, the error δ_k^{l+1} for the third term can be expressed in the same way as in equation (10b) with elements in the $(l+2)$-th layer as follows:

$$\begin{aligned} \delta_k^{l+1} = \frac{\partial C}{\partial z_k^{l+1}} &= \sum_i^{N_{l+2}} \frac{\partial C}{\partial z_i^{l+2}} \frac{\partial z_i^{l+2}}{\partial a_k^{l+1}} \frac{\partial a_k^{l+1}}{\partial z_k^{l+1}} \\ &= \sum_i^{N_{l+2}} \delta_i^{l+2} w_{ik}^{l+2} \sigma'(z_k^{l+1}). \end{aligned} \tag{13}$$

This process continues till the final layer.

In classification problems, the softmax function is commonly used as the activation function in the final layer to normalize the output class probabilities. It provides an effective combination when used in conjunction with the cross-entropy loss function. The cross-entropy loss function can be expressed as follows:

$$C = -\sum_i y_i \log(\hat{y}_i), \tag{14}$$

where y_i is the input truth label, $\hat{y}_i$ is the softmax probability for the i-th class, i.e.,

$$\hat{y}_i = \sigma_s(z_i) = \frac{e^{z_i}}{\sum_k e^{z_k}}. \tag{15}$$

The derivative is

$$\frac{\partial \hat{y}_i}{\partial z_j} = \begin{cases} \hat{y}_i(1 - \hat{y}_j), & \text{if } i = j; \\ -\hat{y}_j \cdot \hat{y}_i, & \text{if } i \neq j. \end{cases} \tag{16}$$

Equation (10a) can be written as

$$\begin{aligned} \delta_j^L = \frac{\partial C}{\partial \hat{y}_i^L} \frac{\partial \hat{y}_i^L}{\partial z_j^L} &= -\sum_i y_i \frac{\partial \log(\hat{y}_i)}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial z_j}, \\ &= -\sum_i y_i \frac{1}{\hat{y}_i} \frac{\partial \hat{y}_i}{\partial z_j}, \\ &= -y_j(1 - \hat{y}_j) - \sum_{i \neq j} y_i \frac{1}{\hat{y}_i}(-\hat{y}_j \cdot \hat{y}_i), \\ &= \hat{y}_j \cdot \sum_i y_i - y_j. \end{aligned} \tag{17}$$

Now, equation (10d) can be written as:

$$\begin{aligned} \frac{\partial C}{\partial w_{jk}^l} &= \delta_j^l a_k^{l-1} \\ &= [\sum_p^{N_{l+1}} \delta_p^{l+1} w_{pj}^{l+1} \sigma'(z_j^l)] a_k^{l-1} \\ &= \{\sum_p^{N_{l+1}} [\sum_q^{N_{l+2}} \delta_q^{l+2} w_{qj}^{l+2} \sigma'(z_j^{l+1})] w_{pj}^{l+1} \sigma'(z_j^l)\} a_k^{l-1} \\ &= \cdots \\ &= \sum_p^{N_{l+1}} \cdots [\sum_i^{N_L} \delta_i^L w_{ij}^L \sigma'(z_j^{L-1})] \times \cdots \times a_k^{l-1} \end{aligned} \tag{18}$$

And for the first layer, it becomes

$$\frac{\partial C}{\partial w_{jk}^1} = \sum_p^{N_{l+1}} \cdots [\sum_i^{N_L} \delta_i^L w_{ij}^L \sigma'(z_j^{L-1})] \times \cdots \times x_k^0 \tag{19}$$

As shown in Equation (18) the gradient calculation involves multiplications and summation. When $\sum_p^{N_{l+1}} \delta_p^{l+1} w_{pj}^{l+1} \sigma'(z_j^l)$ for each layer is greater than 1, the multiplication increases the gradient even more. This can result in gradient exploding problems. On the contrary, when the summation is close to 0, the multiplication makes the gradient even smaller. And gradient vanishing problems could occur.

Authorized licensed use limited to: Utah State University. Downloaded on December 26,2024 at 18:25:55 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2024.3488377

B. Gradient in Convolutional Neural Networks

Modern image recognition tasks leverage CNNs as the primary technique. CNN is a particular variant of feedforward networks comprising three types of layers: convolutional, pooling, and fully connected. Typically, each convolutional layer is followed by a pooling layer for feature learning, and the network culminates with fully connected layers for classification [31]. Without loss of generality, the CNN network consists of two sets of typical CNN layers: two convolutional layers, two pooling layers, and two fully connected layers, which can be readily extended to a more general CNN structure. Moreover, we assume that the dimensions of the layers match, enabling practical training of the CNN. The diagram of this particular CNN is shown in Fig. 3.

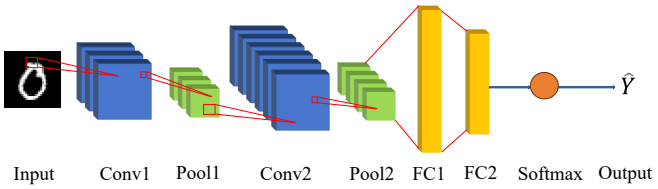


Fig. 3: Convolutional neural network diagram

The feedforward process [32] can be written as:

$$z^1_{j,k} = b^1_{j,k} + \sum_p \sum_q w^1_{p,q} x^0_{j+p,k+q}, \tag{20a}$$

$$a^1_{j,k} = \sigma(z^1_{j,k}), \tag{20b}$$

$$a^2_{j,k} = \max(a^1_{2j,2k}, a^1_{2j+1,2k}, a^1_{2j,2k+1}, a^1_{2j+1,2k+1}), \tag{20c}$$

$$z^3_{j,k} = b^3_{j,k} + \sum_p \sum_q w^3_{p,q} a^2_{j+p,k+q}, \tag{20d}$$

$$a^3_{j,k} = \sigma(z^3_{j,k}), \tag{20e}$$

$$a^4_{j,k} = \max(a^3_{2j,2k}, a^3_{2j+1,2k}, a^3_{2j,2k+1}, a^3_{2j+1,2k+1}), \tag{20f}$$

$$z^5_i = b^5_i + \sum_{j,k} w^5_{i;j,k} a^4_{j,k}, \tag{20g}$$

$$a^5_i = \sigma(z^5_i), \tag{20h}$$

$$z^6_i = b^6_i + \sum_k w^6_{i,k} a^5_k, \tag{20i}$$

$$a^6_i = \sigma(z^6_i). \tag{20j}$$

Now, the backpropagation functions for Fig. 3 are shown in equations (21). In equation (21c), the max-pooling layer is applied with case 1 represented by $a^4_{j,k} = \max(a^3_{2s,2t},\ a^3_{2s+1,2t},\ a^3_{2s,2t+1},\ a^3_{2s+1,2t+1})$, where $s = \frac{j}{2}$ and $t = \frac{k}{2}$. Similarly, in Equation (21d), the 2×2 max-pooling layer is applied.

Here, the softmax function is utilized as the activation function in the final layer, while the cross-entropy function is utilized as the loss function. This setting has found wide applications, such as image classification. There are two kinds of gradients: one in the fully connected layer, i.e., $\frac{\partial C}{\partial w^6_{i,k}}$, $\frac{\partial C}{\partial w^5_{i;j,k}}$, and another in the convolutional layer, $\frac{\partial C}{\partial w^3_{p,q}}$ and $\frac{\partial C}{\partial w^1_{p,q}}$. We will discuss each of them, respectively, in the following section.

$$\delta^6_i = \frac{\partial C}{\partial z^6_i} = \frac{\partial C}{\partial a^6_i}\frac{\partial a^6_i}{\partial z^6_i} = \frac{\partial C}{\partial a^3_6}\sigma'(z^6_i), \tag{21a}$$

$$\delta^5_i = \frac{\partial C}{\partial z^5_i} = \sum_k^{N_6} \frac{\partial C}{\partial z^6_k}\frac{\partial z^6_k}{\partial a^5_i}\frac{\partial a^5_i}{\partial z^5_i},$$

$$= \sum_k^{N_6} \delta^6_k w^6_{ki}\sigma'(z^5_i), \tag{21b}$$

$$\delta^3_{j,k} = \frac{\partial C}{\partial z^3_{j,k}} = \sum_i \frac{\partial C}{\partial z^5_i}\frac{\partial z^5_i}{\partial a^4_{s,t}}\frac{\partial a^4_{s,t}}{\partial z^3_{j,k}},$$

$$= \sum_i \delta^5_i w^5_{i;s,t}\frac{\partial a^4_{s,t}}{\partial a^3_{j,k}}\frac{\partial a^3_{j,k}}{\partial z^3_{j,k}},$$

$$= \sum_i \delta^5_i w^5_{i;s,t}\frac{\partial a^4_{s,t}}{\partial a^3_{j,k}}\sigma'(z^3_{j,k}),$$

$$= \begin{cases} \sum_i \delta^5_i w^5_{i;s,t}\sigma'(z^3_{j,k}), & \text{if case 1;} \\ 0, & \text{otherwise;} \end{cases} \tag{21c}$$

$$\delta^1_{j,k} = \frac{\partial C}{\partial z^1_{j,k}} = \sum_u \sum_v \frac{\partial C}{\partial z^3_{u,v}}\frac{\partial z^3_{u,v}}{\partial a^2_{s,t}}\frac{\partial a^2_{s,t}}{\partial z^1_{j,k}},$$

$$= \sum_u \sum_v \delta^3_{u,v} w^3_{u-s,v-t}\frac{\partial a^2_{s,t}}{\partial a^1_{j,k}}\frac{\partial a^1_{j,k}}{\partial z^1_{j,k}},$$

$$= \sum_u \sum_v \delta^3_{u,v} w^3_{u-s,v-t}\frac{\partial a^2_{s,t}}{\partial a^1_{j,k}}\sigma'(z^1_{j,k}),$$

$$= \begin{cases} \sum_u \sum_v \delta^3_{u,v} w^3_{u-s,v-t}\sigma'(z^1_{j,k}), & \text{if case 2;} \\ 0, & \text{otherwise;} \end{cases} \tag{21d}$$

$$\frac{\partial C}{\partial w^6_{i,k}} = \frac{\partial C}{\partial z^6_i}\frac{\partial z^6_i}{\partial w_i, k^6} = \delta^6_i a^5_k, \tag{21e}$$

$$\frac{\partial C}{\partial w^5_{i;j,k}} = \frac{\partial C}{\partial z^5_i}\frac{\partial z^5_i}{\partial w^5_{i;j,k}} = \delta^5_i a^4_{j,k}, \tag{21f}$$

$$\frac{\partial C}{\partial w^3_{p,q}} = \frac{\partial C}{\partial z^3_{j,k}}\frac{\partial z^3_{j,k}}{\partial w^3_{p,q}} = \delta^3_{j,k} a^2_{j+p,\ k+q}, \tag{21g}$$

$$\frac{\partial C}{\partial w^1_{p,q}} = \frac{\partial C}{\partial z^1_{j,k}}\frac{\partial z^1_{j,k}}{\partial w^1_{p,q}} = \delta^1_{j,k} x^0_{j+p,\ k+q}. \tag{21h}$$

Calculating the gradient $\frac{\partial C}{\partial w^6_{i,k}}$ in the fully connected layer in equation (21e) involves the multiplication of two terms δ^6_j and a^5_k. For $\frac{\partial C}{\partial w^5_{i;j,k}}$,

$$\frac{\partial C}{\partial w^5_{i;j,k}} = [\sum_k^{N_6} \delta^6_k w^6_{ki}\sigma'(z^5_i)]a^4_{j,k}. \tag{22}$$

The summation goes through all neurons in layer 6 and then by multiplication.

For $\frac{\partial C}{\partial w^3_{p,q}}$ and $\frac{\partial C}{\partial w^1_{p,q}}$, we only consider nonzero conditions.

$$\frac{\partial C}{\partial w^3_{p,q}} = \{\sum_i \delta^5_i w^5_{i;s,t}\sigma'(z^3_{j,k})\}a^2_{j+p,\ k+q}$$

$$= \{\sum_i [\sum_k^{N_6} \delta^6_k w^6_{ki}\sigma'(z^5_i)]w^5_{i;s,t}\sigma'(z^3_{j,k})\}a^2_{j+p,\ k+q} \tag{23}$$

Authorized licensed use limited to: Utah State University. Downloaded on December 26,2024 at 18:25:55 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2024.3488377

$$\begin{aligned}\frac{\partial C}{\partial w^1_{p,q}} &= \{\sum_u \sum_v \delta^3_{u,v} w^3_{u-s,v-t} \sigma'(z^1_{j,k})\} x^0_{j+p,\ k+q} \\ &= \{\sum_u \sum_v [\sum_i \delta^5_i w^5_{i;s,t} \sigma'(z^3_{j,k})] w^3_{u-s,v-t} \sigma'(z^1_{j,k})\} \\ &\times x^0_{j+p,\ k+q} \\ &= \{\sum_u \sum_v [\sum_i (\sum_k^{N_6} \delta^6_k w^6_{ki} \sigma'(z^5_i)) w^5_{i;s,t} \sigma'(z^3_{j,k})] \\ &w^3_{u-s,v-t} \sigma'(z^1_{j,k})\} \times x^0_{j+p,\ k+q}\end{aligned} \tag{24}$$

The gradient calculation in convolutional layers also involves multiplication and summation, while the gradient in the front layers involves more arithmetic operations.

C. Gradient in Deep Neural Networks

The summation and multiplication operations accumulate when neural networks deepen and the number of neurons in each layer increases. This can lead to gradient exploding problems [33], where the gradients grow exponentially as they are backpropagated through the neural network from layer to layer. As a consequence, vanilla SGD becomes unstable for model training. Several classical methods have been employed to address the issue of gradient exploding, including gradient clipping, proper weight initialization, and batch normalization. These techniques help control the magnitude of the gradients during the training process. Specifically, gradient clipping is a fast and effective method of limiting gradients to a predefined threshold. The threshold is introduced as an additional hyperparameter that needs to be selected carefully. When the threshold is set too low, the learning ability is restricted with small gradient updates. On the contrary, the clipping effect may be affected if the threshold is too high.

When the error δ^{l+1}_K, the weight w^{l+1}_{kj}, and the derivative of the activation function $\sigma'(z^l_j)$ are small in equation (10b), their summation can result in values in a much smaller range. Typically, the magnitude is much smaller than 1. This situation can lead to the gradient vanishing problem [34], where the gradient exponentially diminishes as it is backpropagated to the earlier layers. Several techniques can be applied to mitigate the risk of gradient vanishing. Proper weight initialization and batch normalization are effective measures to reduce the probability of encountering this problem. However, replacing the sigmoid activation function with a Rectified Linear Unit (ReLU) is the most reliable approach. The ReLU function is defined as follows:

$$\sigma(x) = \begin{cases} 0, & x \leq 0; \\ x, & x > 0. \end{cases} \tag{25}$$

$$\sigma'(x) = \begin{cases} 0, & x \leq 0; \\ 1, & x > 0. \end{cases} \tag{26}$$

This results in the derivative of the activation function in (10b) being 0 or 1. This adjustment enhances the likelihood of the activation function's derivative being 1, contributing to the stabilization of the neural network.

IV. Proposed Method

Motivated by the gradient distribution and effective gradient clipping method to mitigate gradient exploding problems, we first present a received bit masking scheme to restrict the received gradient values within a small range. Then, approximate communication is applied by transmitting gradients with errors. The error level is measured using a l_2-norm. Lower transmission errors would result in better learning performance. Next, to enhance learning performance, the most significant bits (MSBs) in gradients are protected by gray coding. Finally, gradient compression is applied to further reduce communication costs in gradient transmission.

A. Received Bits Masking

Section III provides mathematical demonstrations that establish that gradients can easily cause vanishing or exploding problems under specific conditions. Gradient clipping effectively mitigates the gradient exploding problem. This allows us to establish a threshold for the gradients received by the server. With this in mind, our approach starts by devising a reception mechanism tailored to these gradients.

In ML, gradients are commonly represented by floating-point numbers of 32 bits. These numbers adhere to the structure dictated by the IEEE-754 standard. In this format, the initial bit is allocated for sign information, followed by 8 bits assigned to the exponent, and the concluding 23 bits designated for the fractional component. Each bit is susceptible to noise during transmission, resulting in potential corruption. To avoid block corruption, we integrate interleaving at the transmitter and deinterleaving. This strategy mitigates the probability of multiple error bits aligning together. Within the bit-level representation, if the second bit in the 32-bit form, i.e., the initial bit in the exponent section, is set to 1 while all other 31 bits are 0s, the corresponding decimal value becomes 2. Conversely, when the second bit is 0, and the other 31 bits are 1s, the value resides below 2. Assuming a magnitude threshold of 1 for the gradient value, the exponent's first bit is consistently set to 0. Building upon this insight, on the receiving end, irrespective of the decoded value in the second-bit position of the gradient, we mask it to 0 regardless of the actual received bit value. This adjustment is visually presented in Fig. 4, where the '0' bit signifies the constant value of 0 for that particular bit. In contrast, 'b' adopts the value transmitted during reception.

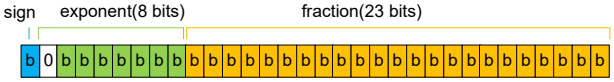


Fig. 4: Received gradient bit representation

B. Approximate Wireless Transmission

With the received bits masking described above, the gradient received at the server falls within the span of $(-2, 2)$, as depicted in Fig. 4. Consequently, the associated error is also limited to a narrow interval. In practical scenarios, this range is often even more minor. In this context, the inherent error

Authorized licensed use limited to: Utah State University. Downloaded on December 26,2024 at 18:25:55 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2024.3488377

resilience of FL systems enables them to train models in the presence of errors. Specifically, gradients in FL systems can be transmitted with errors and need not be perfectly accurate. This resilience to error in FL systems manifests itself in two significant ways. First, the FL system operates within the domain of machine learning and is inherently equipped to manage gradient errors. Secondly, the global gradient aggregation process diminishes the magnitude of the error through averaging. To quantitatively evaluate the impact of errors, the researchers in [35] used relative error to assess individual value discrepancies. This relative error is precisely defined as:

$$E_{rg} = \frac{|g - \tilde{g}|}{g}, \tag{27}$$

where g represents the original gradient value, $\tilde{g}$ denotes the approximated value of g, and E_{rg} stands for the relative error. Given the multitude of gradients present within the neural network, we employ the l_2-norm of errors instead of relative error to evaluate the impact of errors. The l_2-norm of errors is precisely characterized by the following:

$$||E_r||_2 = \sqrt{\sum_{i=1}^{N_g} |g_i - \tilde{g}_i|^2}, \tag{28}$$

where N_g is the number of gradients in the neural network.

We also adopt error tolerance as a metric to assess the threshold that yields comparable learning performance to accurate transmission. In this context, when $||E_r||_2 \leq E_T$, the learning performance remains minimally affected.

C. Most Significant Bit Protection

Furthermore, we have observed that the bits at different locations are of different significance and the gray coding within high-order modulations produces varying effects on bits located at different positions [21]. This observation has significance not only in the transmission of media messages, but also in the transmission of ML models. In wireless communication, the transmission system lacks awareness of the relative importance of data bits and treats all bits equally. For example, when quadrature phase shift keying (QPSK) is used as the modulation scheme, each symbol comprises 2 bits, with possible combinations of $00, 01, 11, 10$. In QPSK, the error probability for the first and second bits is equivalent. In contrast, 16-QAM employs 4 bits per symbol, with a constellation map using gray coding, as illustrated below. As shown in Fig. 5, one of the gray code mappings is depicted. Notably, it becomes evident that the smallest distance between two symbols corresponds to just a 1-bit difference.

In Fig. 5, the underlined bit corresponds to the first bit within each symbol, serving as the most significant bit (MSB) in the context of 16-QAM. Conversely, the fourth or last bit represents the least significant bit (LSB). When symbols share the same transmission probability, the error probability for the MSB is lower than that for the LSB. For example, if the symbol s_0 is decoded with an error, it will most likely be misconstrued as s_1, s_4, or s_5 when the noise power is minimal. For simplicity, we assume that the error symbol can only be

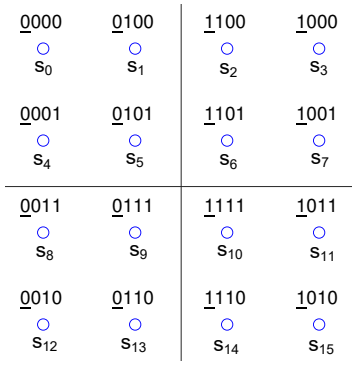


Fig. 5: Gray coding in 16-QAM constellation map

s_1, s_4, s_5. While the MSB remains constant, the LSB changes twice. Assuming that the probability of symbol error from s_0 to s_1 is ρ, then the probability of symbol error for s_4 is ρ, and it becomes $\sqrt{2}\rho$ for s_5. This observation is summarized in Table II. The symbols within the other quadrants exhibit symmetry to that of the first quadrant, yielding identical results. In high-order modulations, gray coding safeguards the MSB bits of gradient values represented in bit form.

TABLE II: Gray coding with 16-QAM MSB/LSB error count

Symbol	Potential Error Symbol	MSB Error Count/ Probability	LSB Error Count/ Probability
s_0	s_1, s_4, s_5	0 / 0	2 / $(1+\sqrt{2})\rho$
s_1	s_0, s_2, s_4, s_5, s_6	2 / $(1+\sqrt{2})\rho$	3 / $(1+2\sqrt{2})\rho$
s_4	s_0, s_1, s_5, s_8, s_9	0 / 0	2 / $(1+\sqrt{2})\rho$
s_5	$s_0, s_1, s_2, s_4, s_6, s_8, s_9, s_{10}$	3 / $(1+2\sqrt{2})\rho$	3 / $(1+2\sqrt{2})\rho$

D. Gradient Compression

Gradient compression can further reduce the communication cost. In [10], we showed that gradient quantization and sparsification can greatly reduce communication time. This would cause a slight loss in final learning performance due to compression errors. In our proposed approximate transmission scheme, there exist transmission errors. We applied the sparsification here. Only the gradients with large gradient magnitude and their indices will be transmitted to the server. Gradient indices will be transmitted accurately through another channel. On the receiver side, the gradients at the specific positions will be set to zero.

E. Summary

The whole process is described in Algorithm 1. Our approach eliminates the need for FEC and packet retransmission. It is important to note that our methodology diverges from UDP, where retransmission is unnecessary. The distinction lies in the fact that UDP operates at a higher level, using the CRC solely to verify the UDP payload. Regarding physical or MAC layer errors, retransmission is still invoked within UDP. In contrast, our approach eliminates FEC and retransmission in the lower layers, encompassing the physical and MAC layers. This approach yields a tripartite benefit: First, it

Authorized licensed use limited to: Utah State University. Downloaded on December 26,2024 at 18:25:55 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2024.3488377

reduces communication overhead, facilitating the transmission of more data bits. Second, it mitigates the computational burden associated with FEC, a particularly attractive feature for edge devices. Third, it enhances latency performance, as retransmission becomes unnecessary.

Algorithm 1 Approximate Wireless Communication for Federated Learning

1: **Initialization: Server** initializes w_0 and broadcasts to all **Clients**.
2: **while** not converge **do**
3: **Client:**
Receives the most recent aggregated global model.
Performs local computation as Eq. (3).
Sends local gradient directly without error correction coding through wireless channels.
4: **Server:**
Receives the local gradient with errors.
Masks the second bit in the 32-bit gradient representation as 0, leaving the other bits as received.
Sends the aggregated global model to clients.
5: **end while**

V. Simulation Results

This section begins by presenting the configurations of the simulation parameters. After that, we conducted BER simulation under different modulation schemes. Notably, among the modulation schemes, the lower-order modulation technique exhibits a superior BER to higher-order modulation, given the same SNR level. Specifically, QPSK outperforms 16-QAM and 256-QAM in terms of BER. Moving forward, we can compare FL performance across three scenarios: ECRT, naive erroneous transmission, and erroneous message augmented by our proposed strategy. The naive erroneous transmission signifies that transmission in wireless networks is subject to errors without supplementary measures. In contrast, our proposed scheme shows significantly improved testing accuracy compared to naive erroneous transmission. Compared to ECRT transmission, our proposed method yields substantial time savings while achieving commendable performance. The convergence of our proposed scheme is proved using three different image datasets in both IID and non-IID data settings. Then, the aggregation error is quantized and the error is evaluated under different wireless conditions. Furthermore, we dive into the implications of gray coding across various modulation schemes, highlighting the innate bit protection for MSBs. Finally, we investigate the influence of user participation numbers on the FL process. As user participation increases, aggregation errors decrease, indicating a pronounced improvement.

Our simulation considers a prototypical FL setup with $M = 100$ clients that interact with the server. We leverage three image datasets to validate our proposed approach: MNIST, Fashion-MNIST, and Cifar-10. These datasets encompass 10 distinct classes, and while MNIST and Fashion-MNIST entail grayscale images, Cifar-10 features color images. Specifically, MNIST comprises handwritten digits from 0 to 9, Fashion-MNIST showcases fashion articles like shoes and clothing, and Cifar-10 encompasses images of vehicles and animals. MNIST and Fashion-MNIST provide 60,000 images for training, whereas Cifar-10 offers 50,000. The test subsets for all datasets contain 10,000 images. Our simulation encapsulates Independently and Identically Distributed (IID) and non-IID data scenarios. In the IID setup, each client randomly samples a fixed number of images from the training data. Conversely, in the non-IID setup, each client is allocated images from 2 specific classes. Across the three datasets, CNNs are employed with varying architectures. The CNN configuration is similar to the datasets, incorporating 2 convolutional layers with 5-unit kernels, 2 max-pooling layers with a size of 2, and multiple fully connected layers in conclusion. The activation function in all layers, except the final one, uses ReLU, while the last layer uses the log-softmax function. In particular, the neural network architectures in the MNIST, Fashion-MNIST, and Cifar-10 datasets encompass 21,840, 65,558, and 62,006 model parameters, respectively. The learning rate is $\eta = 0.01$, and the size of the training batch is 10. Throughout the learning process, testing follows each round after global aggregation, covering all data samples in the test data set. All clients participate in the training procedure, unless otherwise specified.

In the communication model, we establish the path loss exponent for the wireless channel as $\alpha = 3$ and assume a distance of 10 meters between the server and the clients. The transmission power at the clients is normalized to 1.

QPSK achieves the best BER performance. The BER is summarized in Table III.

TABLE III: BER summary

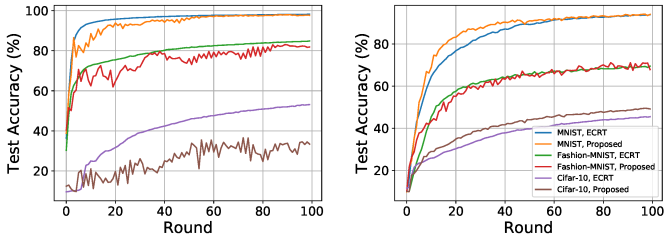
SNR	QPSK	16-QAM	256-QAM
0 dB	2.11×10^{-1}	3.28×10^{-1}	4.26×10^{-1}
10 dB	4.36×10^{-2}	1.23×10^{-1}	2.79×10^{-1}
20 dB	4.91×10^{-3}	1.90×10^{-2}	1.12×10^{-1}

To begin with, we demonstrate that the proposed scheme attains a convergence point similar to that of the ECRT scheme under the high SNR regime while experiencing a marginal reduction in performance under the low SNR regime. Fig. 6 illustrates the learning outcomes for three different data sets, comparing the transmission of ECRT with our proposed transmission method using QPSK modulation. We use the IID data set in Fig. 6(a). When our proposed method is used, the learning curve exhibits a more significant fluctuation than that observed with ECRT transmission. This variance is attributed to transmission errors introduced by approximate communication. However, except for the Cifar-10 dataset, the convergence point of learning of our proposed method aligns closely with that of ECRT transmission for the other two datasets. This fluctuation becomes more streamlined in the high SNR regime shown in Fig. 6(b). Furthermore, due to the non-IID data structure, our proposed method showcases improved learning performance in certain instances, surpassing that of ECRT transmission.

Wi-Fi transmission nominally targets the SNR range of 10-30 dB [36]. When SNR=0 dB, there will be no valid packet due

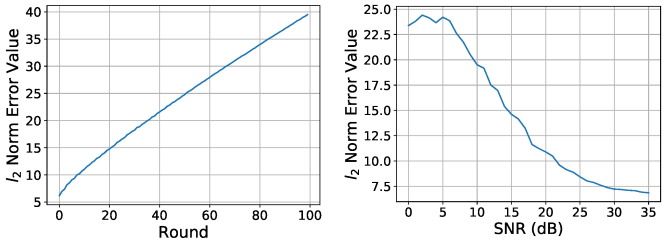
Authorized licensed use limited to: Utah State University. Downloaded on December 26,2024 at 18:25:55 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2024.3488377



(a) Test accuracy of IID Data at SNR=10 dB (b) Test accuracy of non-IID Data at SNR=20 dB

Fig. 6: Test accuracy



(a) l_2 error norm at SNR =10 dB (b) Average l_2 error norm at different SNR

Fig. 8: l_2 error norm

to transmission errors and packet retransmission. Our proposed method still achieves about 60% test precision when SNR = 0 dB, as shown in Fig. 7. The proposed method is superior to the ECRT method with a very low SNR. In addition, a high SNR gets better test accuracy than a low SNR.

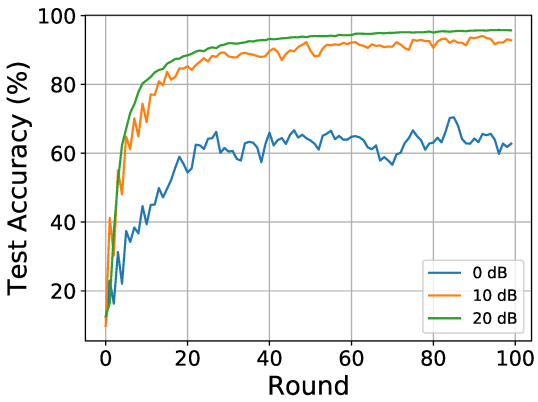


Fig. 7: Test accuracy of MNIST, non-IID with different SNR

As shown in Table III, the BER decreases with SNR increasing in all modulation schemes. To quantify the impact of errors, we employ the l_2-norm of the gradient aggregation error as our evaluation metric, as depicted in Equation (28). Fig. 8(a) shows the error values of the norm gradient l_2 when using our proposed method with 256-QAM in the MNIST dataset under non-IID conditions at SNR=10 dB. In particular, the error exhibits a linear accumulation pattern. To assess the influence of SNR on the aggregation error, we employ average aggregation as our measure. For each level of SNR, we run 100 rounds and calculate the average value of the l_2 norm error. Fig. 8(b) illustrates that the average error decreases with higher SNR values. This observation explains why the learning performance is superior at SNR=20 dB compared to SNR=10 dB, as demonstrated in Fig. 6 and Fig. 7.

Within the ECRT scheme, reliable bit reception at the server is ensured through error correction coding and retransmission mechanisms, though at the expense of time and energy. Conversely, the naive and erroneous transmission approach involves transmitting flawed bits without gaining insight into gradient values. This leads to a test accuracy that remains flat at approximately 10%, akin to random guessing, as depicted in Fig. 9. On the other hand, our proposed method leverages prior knowledge of gradient values, which are anticipated to be distributed in a small range. As a result, our proposed scheme outperforms naive error transmission, resulting in significantly improved outcomes.

To quantify the time savings achieved by our proposed method compared to ECRT transmission, we employ a practical IEEE 802.11 protocol coupled with LDPC error correction coding. Compared with the Different schemes run on the same platform. The test accuracy versus relative running time serves as the performance metric. LDPC is a promising error correction code that approaches the Shannon limit. The choice of coding rate introduces a trade-off between error correction capability and transmission overhead. Lower coding rates increase transmission overhead, but offer enhanced error correction capabilities. Our approach employs a coding rate of $1/2$, which is used to improve error correction. As reported in [37], for a code rate of $1/2$ with a code length of 648 and the use of the parity check matrix for the search, the minimum Hamming distance amounts to 15. This translates to an error correction capacity of 7 bits. ECRT transmission using QPSK modulation entails retransmission rates of 3.4% and 23% at SNR=20 dB and SNR=10 dB, respectively. Retransmission of the packet will reduce TCP throughput [38]. As demonstrated in Fig. 9, the use of LDPC coding and retransmission extends the time required to achieve test precision 80% to twice that of the proposed scheme at SNR=20 dB. Consequently, this difference increases to more than threefold at SNR = 10 dB, highlighting the advantage of the proposed method. The time along the x-axis is the relative time that different schemes run on the same platform. When running these algorithms on another platform, the absolute time might change, but the relative time keeps.

To show the effectiveness of inherent MSB bit protection offered by gray coding in high-order modulation, we start by presenting the test accuracy of distinct modulations at an equivalent SNR in Fig. 10(a). Fashion-MNIST attains comparable learning performance when modulated using QPSK, 16-QAM, and 256-QAM, all at an SNR of 10 dB. This underlines the resilience conferred by gray coding in varying modulation schemes.

Table III presents the target SNR values required for different modulations to obtain an identical BER. Although QPSK, 16-QAM, and 256-QAM achieve similar BER levels as depicted in Fig. 10(b), it is noteworthy that the learning

Authorized licensed use limited to: Utah State University. Downloaded on December 26,2024 at 18:25:55 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2024.3488377

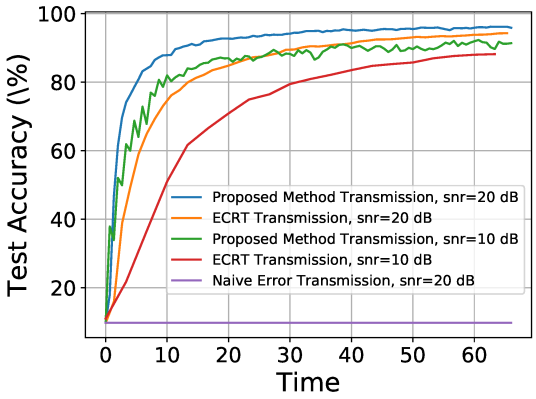
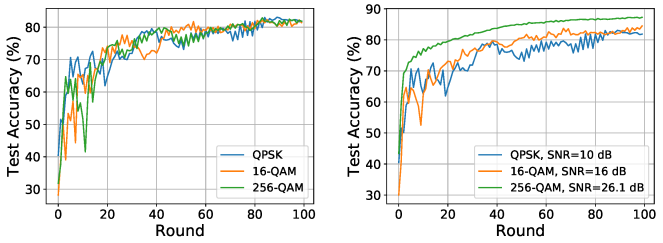


Fig. 9: Test accuracy of MNIST under non-IID

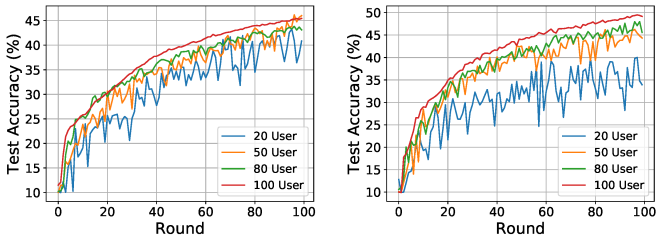


(a) Test accuracy at the same SNR=10 dB

(b) Test accuracy at the same BER $\approx 4.36 \times 10^{-2}$

Fig. 10: Test Accuracy of Fashion-MNIST under IID

performance of gray-coded 256-QAM transmission surpasses that of both QPSK and 16-QAM. This observation underscores the enhanced performance obtained by employing gray coding in conjunction with 256-QAM modulation.



(a) Test accuracy using ECRT transmission

(b) Test accuracy using our proposed method

Fig. 11: Test accuracy of Cifar-10 under non-IID

In the context of FL, model averaging is employed for aggregation, which inherently mitigates error levels. As demonstrated in Fig. 11(a), the participation of 20, 50, 80, or 100 users in the learning process, free of errors, produces comparable learning outcomes. In contrast, using QPSK modulation at SNR=20 dB in Fig. 11(b), when 20 users engage in learning with errors introduces more pronounced fluctuations. However, when the participant count increases to 50, 80, or 100, the behavior mirrors that of the ECRT transmission. Comparing the learning performance between ECRT and our proposed method, the difference when 100 users participate in the learning is smaller than 20 users. This means that by involving more users in our proposed method, the expected outcome is reduced error, and the learning performance is improved due to the error averaging.

TABLE IV: SNR for target BER

BER	QPSK	16-QAM	256-QAM
4.36×10^{-2}	10 dB	16 dB	26.1 dB
4.91×10^{-3}	20 dB	26.1 dB	36.5 dB

The effectiveness of gradient compression is shown from the perspective of the number of required rounds. The gradient compression takes place on the clients. Compared with computation, communication is the bottleneck in wireless networks. The FL running time here only considers the local model uploading time and global model downloading time. The local computation time and compression time are not considered. When the gradients are sparsified to 10%, 30%, 50%, 70%, 90%, respectively, the final learning performance is shown in Fig. 12. From the results, when only 10% of the gradients are kept, the best learning performance is achieved. In approximate communication, the gradients with errors can also be destructive rather than constructive. From a time perspective, when the sparsification rate is 10%, while the indices of the kept gradients need to be transmitted in accurate channels, compression still saves close to $10\times$ time.

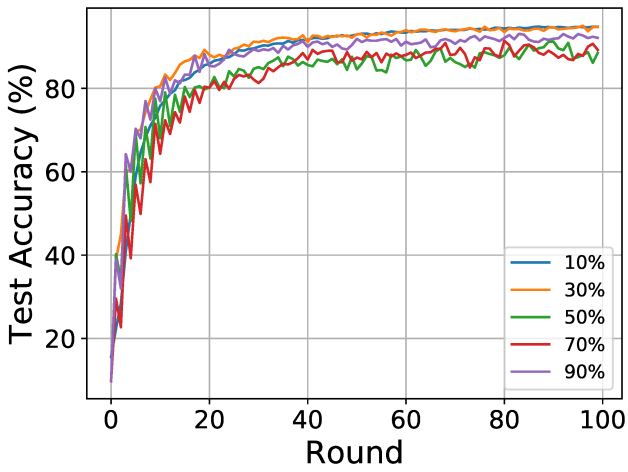


Fig. 12: Test accuracy of MNIST under non-IID with approximate communication and sparsification

VI. Conclusion

This paper introduces a novel approach to transmitting model parameters in federated learning within wireless networks for resource-constrained IoT devices. Unlike existing transmission methods that rely on forward error correction and retransmission, our proposed scheme involves gradient transmission with errors leveraging prior knowledge of the gradient values. This knowledge is based on the gradient distribution. By masking the bits received, our approach achieves significantly improved learning performance with errors, outperforming naive error transmission. In particular, it achieves learning performance comparable to conventional transmission

Authorized licensed use limited to: Utah State University. Downloaded on December 26,2024 at 18:25:55 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2024.3488377

with forward error correction and packet retransmission, while consuming at least half the time. Furthermore, this study investigates the impact of factors such as the variance of the aggregation error with SNR, the effectiveness of gray coding in high-order modulation, and the influence of the number of users participating in the learning process under error conditions. Gradient sparsification can further reduce communication costs and mitigate the side effects caused by approximate gradient transmission.

VII. Acknowledge

The work of H. Sun is supported by UGA E-mobility Initiative and NSF CNS-2236449. The work of R. Q. Hu is supported by NSF EEC-1941524, ECCS-2139508, and CNS-2319487. The work of Y. Qian is supported by NSF ECCS-2139520 and CNS-2319486.

References

[1] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," [Online]: https://arxiv.org/abs/1610.05492, 2016.

[2] B. Brik, A. Ksentini, and M. Bouaziz, "Federated learning for UAVs-enabled wireless networks: Use cases, challenges, and open problems," *IEEE Access,* vol. 8, pp. 53841–53849, Mar. 2020.

[3] M. Shirvanimoghaddam, A. Salari, Y. Gao, and A. Guha, "Federated learning with erroneous communication links," *IEEE Commun. Lett.*, vol. 26, no. 6, pp. 1293-1297, Apr. 2022.

[4] A. Nafaa, T. Taleb, and L. Murphy, "Forward error correction strategies for media streaming over wireless networks," *IEEE Commun. Mag.*, vo. 46, no. 1, pp. 72-79, 2008.

[5] M. F. Tsai, N. Chilamkurti, C. K. Shieh, and A. Vinel, "Mac-level forward error correction mechanism for minimum error recovery overhead and retransmission." *Mathematical and Computer Modeling*, vol. 53, no. 11-12, pp. 2067-2077, 2011.

[6] I. Ez-Zazi, M. Arioua, El Oualkadi, and Y. el. Assari, "Joint FEC/CRC coding scheme for energy constrained IoT devices," *Proceedings of the International Conference on Future Networks and Distributed Systems*m pp. 1-8, 2017.

[7] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and H. A. Arcas, "Communication-efficient learning of deep networks from decentralized data," [Online]: https://arxiv.org/abs/1602.05629, 2016.

[8] M. Chen, H.V. Poor, W. Saad, and S. Cui, "Convergence time optimization for federated learning over wireless networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 4, pp. 2457-2472, 2020.

[9] X. Ma, H. Sun, Q. Wang, R.Q. Hu, "Scheduling policy and power allocation for federated learning in NOMA based MEC," in *IEEE Globecom*, pp. 1-7, 2020.

[10] H. Sun, X. Ma, R. Q. Hu, "Adaptive federated learning with gradient compression in uplink NOMA," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 16325-16329, 2020.

[11] F. Seide, H. Fu, J. Droppo, G. Li and D. Yu, "1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[12] A. F. Aji, and K. Heafield, "Sparse communication for distributed gradient descent," [Online]: https://arxiv.org/abs/1704.05021, 2017.

[13] Z. Yang, M. Chen, W. Saad, C.S. Hong and M. Shikh-Bahaei, "Energy-efficient federated learning over wireless communication networks," *IEEE Trans. Wireless Commun.*, vol 20, no. 3, pp. 1935-1949, 2020.

[14] F. Betzel, K. Khatamifard, H. Suresh, D. J. Lilja, J. Sartori, U. Karpuzcu, "Approximate communication: Techniques for reducing communication bottlenecks in large-scale parallel systems." *ACM Comput. Surv.*, vol. 51, no. 1, pp. 1-32, Jan. 2018.

[15] 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*, vol. 30, 2017.

[16] A. M. Abdelmoniem, A. Elzanaty, M. S. Alouini, and M. Canini, "An efficient statistical-based gradient compression technique for distributed training systems," in *Proceedings of Machine Learning and Systems*, vol. 3, pp. 297-322, 2021.

[17] B. Guo, Y. Liu, and C. Zhang, "A partition based gradient compression algorithm for distributed training in aiot," in *Sensors*, vol. 21, no. 6, pp. 1943, 2021.

[18] F. Sunny, A. Mirza, I. Thakkar, M. Nikdast, and S. Pasricha, "ARXON: A framework for approximate communication over photonic networks-on-chip," [Online]: https://arxiv.org/abs/2103.08828, 2021.

[19] Y. Chen, and A. Louri, "An approximate communication framework for network-on-chips," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no.6, pp. 1434-1446, 2020.

[20] M. F. Reza, and P. Ampadu, "Approximate communication strategies for energy-efficient and high performance NoC: Opportunities and challenges," *Great Lakes Symposium on VLSI 2019*, May 2019.

[21] S. Sen, S. Gilani, S. Srinath, S. Schmitt, and S. Banerjee, "Design and implementation of an "Approximate" communication system for wireless media applications," *Proceedings of the ACM SIGCOMM 2010 Conference*, pp.15-26, 2010.

[22] O. Shamir, N. Srebro, and T. Zhang, "Communication-efficient distributed optimization using an approximate newton-type method," *Proceedings of the 31st International Conference on Machine Learning*, vol. 32, no. 2, pp. 1000-1008, PMLR, 2014.

[23] X. Su, Y. Zhou, L. Cui, and J. Liu, "On model transmission strategies in federated learning with lossy communications," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 4, pp. 1172-1185, 2023.

[24] R. Y. Choi, A. S. Coyner, J. Kalpathy-Cramer, M. F. Chiang, and J. P. Campbell, "Introduction to machine learning, neural networks, and deep learning," *Translational Vision Science & Technology*, vol. 9, no. 2, Jan. 2020.

[25] M. A. Nielsen, "Neural networks and deep learning", *Determination Press*, 2015.

[26] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," [Online]: https://arxiv.org/abs/1811.03378, 2018.

[27] I. Goodfellow, Y. Bengio, and A. Courville, "Deep learning," *MIT Press*, 2016.

[28] I. Sutskever, J. Martens, G. Dahl, G. Hinton, "On the importance of initialization and momentum in deep learning," in *International Conference on Machine Learning,* pp. 1139-1147, PMLR, 2013.

[29] X. Glorot, and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 249-256, 2010.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1026-1034, 2015.

[31] K, O'Shea, and R. Nash, "An introduction to convolutional neural networks," [Online]: https://arxiv.org/abs/1511.08458, 2015.

[32] P. Murugan, "Feed forward and backward run in deep convolution neural network," [Online]: https://arxiv.org/abs/1711.03278, 2017.

[33] G. Philipp, D. Song, J. G. Carbonell, "The exploding gradient problem demystified - definition, prevalence, impact, origin, tradeoffs, and solutions," [Online]: https://arxiv.org/abs/1712.05577, 2018.

[34] H. H. Tan, and K. H. Lim, "Vanishing gradient mitigation with deep learning neural network optimization," *2019 7th International Conference on Smart Computing & Communications (ICSCC)*, pp. 1-4. IEEE, 2019.

[35] Y. Chen, and A. Louri, "Learning-based quality management for approximate communication in network-on-chips," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 39, no. 11, pp. 3724-3735, 2020.

[36] M. Duarte, A. Sabharwal, V. Aggarwal, R. Jana, K.K. Ramakrishnan, C.W. Rice, and N.K. Shankaranarayanan, "Design and characterization of a full-duplex multiantenna system for WiFi networks," *IEEE Trans. Vehi. Techno.*, vol. 63, no. 3, pp. 1160-1177, 2013.

[37] B. K. Butler, "Minimum distances of the QC-LDPC codes in IEEE 802 communication standards," [Online]: https://arxiv.org/abs/1602.02831, 2016.

[38] S. Varma, "Chapter 4 - Congestion control in broadband wireless networks," *Internet Congestion Control,* ISBN 9780128035832, pp. 103-134, https://doi.org/10.1016/B978-0-12-803583-2.00004-9.

Authorized licensed use limited to: Utah State University. Downloaded on December 26,2024 at 18:25:55 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.