HCFL: A High Compression Approach for Communication-Efficient Federated Learning in Very Large Scale IoT Networks

Minh-Duong Nguyen, Sang-Min Lee, Quoc-Viet Pham, Member, IEEE, Dinh Thai Hoang, Senior Member, IEEE, Diep N. Nguyen, Senior Member, IEEE, and Won-Joo Hwang, Senior Member, IEEE

Abstract—Federated learning (FL) is a new artificial intelligence concept that enables Internet-of-Things (IoT) devices to learn a collaborative model without sending the raw data to centralized nodes for processing. Despite numerous advantages, low computing resources at IoT devices and high communication costs for exchanging model parameters make applications of FL in massive IoT networks very limited. In this work, we develop a novel compression scheme for FL, called high-compression federated learning (HCFL), for very large scale IoT networks. HCFL can reduce the data load for FL processes without changing their structure and hyperparameters. In this way, we not only can significantly reduce communication costs, but also make intensive learning processes more adaptable on low-computing resource IoT devices. Furthermore, we investigate a relationship between the number of IoT devices and the convergence level of the FL model and thereby better assess the quality of the FL process. We demonstrate our HCFL scheme in both simulations and mathematical analyses. Our proposed theoretical research can be used as a minimum level of satisfaction, proving that the FL process can achieve good performance when a determined configuration is met. Therefore, we show that HCFL is applicable in any FL-integrated networks with numerous IoT devices.

Index Terms—Autoencoder, communication efficiency, data compression, deep learning, distributed learning, federated learning, internet-of-things, machine type communication

1 INTRODUCTION

The rapid increase in Internet-of-Things (IoT) applications has revolutionized the Big Data and machine learning (ML) technology fields [1]. Conventionally, ML and Big Data algorithms are deployed in the centralized servers [2]. However, communication and computing costs are not scalable with the number of IoT devices, which has limited the use of ML solutions in IoT. FL [5], an emerging AI paradigm, has recently attracted considerable attention from the research communities thanks to its nice features and wide-ranging applications. In conventional ML techniques, a server collects the user data and performs a computing process centrally, thereby increasing the risk of high communication overload and data leakage. FL, on the other hand, aims to mitigate the privacy concerns by locally training the ML models, and thus the users only need to dispatch to the server their model parameters [6]. Therefore, this paradigm preserves the private and collaborative aspects of the ML framework for distributed users. As a result, a number of AI applications leveraging the FL process for IoT networks have been developed recently, such as vehicular cloud [7], remote applications [8], smart healthcare [9], and mobile edge networks [10].

Despite recent advancements in computing hardware and computing paradigms such as fog computing and mobile edge computing (MEC), the limitation of communication resources still remains as an obstacle in IoT systems [11]. With
the fact that the number of IoT devices is increasing dramatically in the modern world and the wireless resources are limited, communication proficiency sets off to be one of the key challenges for carrying out massive IoT scenarios in which the large-scale FL system is integrated. Therefore, some approaches to address the communication efficiency problem for FL processes have been introduced recently [12]. For example, the authors in [13] utilize the submodel framework, wherein the clients decide which parts are necessary for the model update process instead of implementing the whole model. The FedBoost algorithm in [14] is a combination of an ensemble learning and the random sampling theory. To be more specific, FedBoost selects a random partial set of clients to deliver the updates instead of sending all parameters to every client in each round. The secure aggregation approach proposed in [15] and [16] utilizes the random sampling method to cross-validate the model updates between different pairs of clients before forwarding the model parameters to the aggregation module at the server. Furthermore, the authors in [17] and [18] formulate the optimization problems to minimize the time cost for each communication round [17] and maximize the aggregation data rate from FL users [18]. In order to reduce the communication load during the propagation phase, many researches have applied sparsification and model pruning into FL. Authors in [19] inherit the simple regularization method to develop pruneFL algorithm. This algorithm can reduce the number of parameters up to 7 times. PruneFL, on the other hand, requires more than 6000 communication rounds to reach convergence, which is inappropriate for IoT networks. CA-DSDG [20] and CE-FedAvg [21] both employ the same sparsification concept, in which model parameters that do not change much are dropped from the updating phases. However, the efficiency by sparsification is not convincing where the achieved compression ratio is capped at 70%. Another effective method which is both efficient for communication and computation is model quantization. The works in [22], [23], [24] focus on model compression wherein the entire network structure is estimated and incorporated into a new model with a low definition model parameter set. The full-precision weights are estimated into approximate set with ternary format (which includes only three values: −1, 0, 1) [25]. Since the deep network parameters are quantized into 2-bit values, the method already reaches its compression limitation. Therefore, the maximum compression efficiency that this method can achieve is constrained by 90%.

In order to further improve the communication efficiency of FL processes, in this work, we propose a new compression strategy, called high-compression federated learning (HCFL), for IoT networks with a very large number of IoT devices. Remarkably, thanks to the compact design on the presentation layer of the OSI model [26], HCFL is strictly in compliance with the entity-specific information flow between current 3GPP interfaces while being communication efficient. Thus, our proposed HCFL scheme can be cooperated with other communication-efficient methods (e.g., quantization [22], [23], [27], sparsification [20], model pruning [19], [21]) to improve the performance of the whole FL process. We can sum up our contributions in this paper as follows:

- We introduce a novel compression scheme, called HCFL, for FL. We leverage an under-complete autoencoder structure in the FL process to develop a communication-efficient strategy that can reduce the transmission load of the FL mega-system with a very large number of IoT devices.
- We provide theoretical analysis of the proposed HCFL method under the FL mega-system. We demonstrate that the HCFL method can achieve a very high performance, e.g., the models with HCFL can be compressed more than 95% of their original size. We also prove that the lossy compression by HCFL adds noise into the FL training process, which encourages exploration efforts for our large-scale IoT learning system. Thus, HCFL can improve the FL’s overall performance and reduce FL’s both convergence round and delays. Furthermore, we investigate a relationship between the system performance and the other compress-aided FL process’s characteristics such as the number of IoT devices and the compressor reconstruction error.
- We conduct various simulations with an in-depth comparison with the conventional FL process in terms of performance and communication efficiency. We evaluate the proposed HCFL method under different conditional simulations, including compression efficiency at different compression ratios, convergence analysis at distinct FL process and client predictor settings. The simulation results reveal that the HCFL method can achieve a compression ratio up to 32 times better than that of the conventional FL process. Notably, the trade-off between compression ratio and performance is acceptable when the accuracy loss on the HCFL system is less than 3% in comparison with the baseline method.

The rest of this article is organized as follows. Section 2 briefly reviews the fundamental knowledge about FL and autoencoder concept. Section 4 demonstrates the theoretical analysis of HCFL system and how the HCFL and FL process support each other. Section 3 details the deployment strategy of the compression-aided federated system along with two appropriate compression model structures. Simulations conducted to compare the performances of a standard FL process and the proposed method in terms of global loss, data reconstruction error and convergence rate are described in Section 6. Finally, conclusions are presented in Section 7.

2 BACKGROUND AND METHODS

This section first presents the system model, then provides some preliminaries of the conventional FL procedure and its essential formulations, and finally shows the main features and definitions of the autoencoder model.

2.1 System Model

We consider a mega FL-integrated IoT system involving one server and a set of $K$ clients (i.e., IoT devices), as illustrated in Fig. 1. Clients and server jointly build a shared ML model for processing and data analysis. Each client collects data from its sensors or users. We assume that all the clients’ datasets have the IID property in this study [5]. The client’s data is processed through an ML-based predictor on each device distributively. Because a large number of clients participate in each communication round, every participating client needs to share limited bandwidth resources. Thus, the transmission
rate of each client is also limited. This problem leads to the need for communication efficiency in FL-integrated IoT networks and motivates us to complete this work. Our work aims to reduce the communication burden significantly in the meta FL-integrated IoT system.

2.2 Federated Learning

FL is an up-to-the-minute concept of ML. In FL, instead of collecting the data and utilizing them for central training at one location, the data is processed distributedly at the clients. The cooperation between the server and clients is to ensure that the model losses are verified at the server, and the improvement is updated onto the distributed clients gradually, as shown in Fig. 1. The FederatedAveraging (FedAvg) algorithm, proposed in [5], utilizes a sum averaging function to combine loss and weight values from different clients as follows:

$$ f(w) = \frac{n_k}{n} \sum_{k=1}^{K} F_k(w) \quad \text{where} \quad F_k(w) = \frac{1}{n_k} \sum_{i \in F_k} f_i(w), $$

where, \( w \) denotes the parameters of the FL model, \( F_k(w) \) and \( f_i(w) \) denote the loss value on the client \( k \) and each data point of the whole dataset with a size \( n \), respectively. The dataset is distributed over \( K \) clients with a data size of \( n_k \) for each client in FL. The gradient on the client \( k \) is given as \( g_k = \delta F_k(w_k) \).

The aggregation function at the server generates a close-form function with the loss sum-averaging update as follows:

$$ w_{t+1} = \frac{\sum_{k=1}^{K} n_k}{n} w_t^{k+1}. $$

The equality \( \frac{n_k}{n} = K \) is valid when the dataset on each client is IID with each other on the whole set of dataset of the system [28] and every client samples the same amount of data each communication round. Thus, the FedAvg algorithm updates the model weights as follows:

$$ w_{t+1} = \frac{1}{K} \sum_{k=1}^{K} w_t^{k+1}. $$

2.3 Autoencoder

Autoencoders utilized in deep learning [2] are dual-symmetric neural networks designed to produce the output data that is approximately-but-not-equal to the intake of that system. Autoencoders learn a concentrated representation of the input while the most valuable information is retained. There are two components in the aforementioned system: an encoder that processes a transformation: \( h = f_o(x) \), and a decoder \( \hat{x} = g_o(h) \) that is responsible for the reconstruction. The \( f \) and \( g \) represent the encoder and decoder neural networks, respectively, and \( h \) denotes the feature vector of the model in the aforementioned functions. The encoder transforms the incoming matrix into a completely divergent model, whereas the decoder \( g \) is tasked with rehabilitation. The parameters of the proposed network is learned simultaneously on the task of reconstructing the original output.

The loss function \( L(w) \), which is a measure of the discrepancy between the input \( x \) and the output \( \hat{x} \), is applied to obtain the lowest possible reconstruction error [29], which is defined as follows:

$$ L(w) = \frac{1}{2} \| \hat{x} - x \|^2 = \frac{1}{2} \sum_{k=1}^{N} (\hat{x} - x)^2. $$

The proposed back-propagation operation uses the conventional mean squared error (MSE) loss function [2] to compute the euclidean distance between the two vectors of the output and input of the autoencoder.

The illustration of a general autoencoder is shown in Fig. 2. Autoencoders are designed to not copy the output from the input perfectly, as aforementioned. Usually, restrictions are conventionally imposed to make the distribution of the output data to be exactly the same with that of the original information. Because of the objective to prioritize the input aspects, which can represent the whole data distribution, the encoder manages to collect the valuable properties only. The main objectives of autoencoders are conventionally feature extraction, dimensionality reduction, and data augmentation. Autoencoders learn the data characterization in a reduced dimensional space by placing additional focus on essential features while attempting to eliminate redundancy and noise. It is based on the encoder-decoder architecture, wherein the encoder encrypts the data with a high dimensional space to the lower-dimensional version. Moreover, the decoder performs in a reverse way, i.e., the decoder reconstructs the initial high-dimensional information from the reduced-dimensional data.

3 PROPOSED COMPRESSION-AIDED FL ALGORITHM

In this section, we present how to implement our proposed HCFL-integrated FL in massive IoT networks. We
first propose the problem formulation for our HCFL method. Then, we propose the system deployment of HCFL in an FL process and clarify the procedure of the HCFL-aided FL. Second, we find that when implementing a traditional multi-layered autoencoder, the performance of the HCFL model is degraded due to the curse of dimensionality. Therefore, we propose new model architectures for the compressor and extractor that improve the integration efficiency of the HCFL into the FL process. Furthermore, we reveal how to pre-process the input model weights and how the model is trained given the model parameter dataset.

### 3.1 Problem Formulation

In this part, we aim to develop the framework to minimize the reconstruction error between the original model at input and reconstructed model at output for HCFL in the FL process. In general, HCFL operates as an autoencoder to compress the distributed clients’ model and reconstruct the encoded model. Nevertheless, because the clients apply gradient descent stochastically, the models’ weight distribution transformation remains undercover. To acknowledge those problems, we need to capture the generalization and transforming trends of every model’s distributions, especially when the model’s gradient descent is operated in different directions. Thus, our proposed HCFL should be satisfied the two following conditions:

- Minimize the reconstruction error between the original model and reconstructed model as mentioned in (4).
- Maximize the mutual information between the original model and encoded data. Therefore, we can maximize the information transferred from the original data to the encoded data, which improves the performance of the encoder and thus enhance the decoder quality. The optimization problem is introduced as follows:

\[
\max_{\theta} I(W, C) = D_{KL}[p(W, C) \| p(W) \otimes p(C)],
\]

where \(\otimes\) is the Kronecker product between two matrices \(W\) and \(C\), \(\theta\) is the model parameters of the HCFL model, and the \(D_{KL}\) stands for relative entropy between two data distributions.

In this way, we can formulate a joint optimization problem to solve both of the aforementioned tasks. Meanwhile, to conjugate the distance-based problem in (4) and entropic problem in (5), we consider the relationship between MSE and cross entropy (CE). We follow an assumption that the output of deep network in HCFL is demonstrated as Gaussian distribution with variance of \(\sigma^2\):

\[
p(\tilde{W} | W) = \mathcal{N}(W, \sigma^2)
\]

\[
= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2} \frac{(W - \tilde{W})^2}{\sigma^2}\right).
\]

Applying CE on the HCFL’s output, we have:

\[
H(W, \tilde{W}) = -E_{p(\tilde{W})} \log p(\tilde{W})
\]

\[
= -E_{p(\tilde{W})} \log \left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2} \frac{(W - \tilde{W})^2}{\sigma^2}\right)\right]
\]

\[
= O(-E_{p(\tilde{W})} (W - \tilde{W})^2).
\]

\(O\) describes the growth rate of the function, \(E_{p(\tilde{W})}\) is the expectation over distribution of model parameters \(W\). From the above formula, we can see that \(H(W, \tilde{W})\) is proportional with MSE loss \(E_{p(\tilde{W})} (W - \tilde{W})^2\). Thus, we have the joint problem formulation as follows:

\[
\min_{\theta} L = H - I
\]

\[
s.t. \quad H = \lambda H(W, \tilde{W}),
\]

\[
I = (1 - \lambda) D_{KL}[p(W, C) \| p(W) \otimes p(C)],
\]

where \(\theta\) is the parameter set of the HCFL model and \(\lambda\) is the scale coefficient between two tasks \(H\) and \(I\). Intuitively, the problem in (8) is the alternative version of bottleneck technique. Therefore, the choice of \(\lambda\) is similar to the scaling factor choice in [30], [31]. The HCFL is thus trained by the following gradient descent process:

\[
\theta = \theta - \gamma \nabla L,
\]

where \(\gamma\) is the learning rate for the HCFL training process.

### 3.2 System Deployment

This part discusses the system deployment of the HCFL in the FL. As we can see from Fig. 3, the HCFL comprises two components: encoders which are located on each client’s domain, and a single decoder which is embedded in the server’s firmware. Although numerous encoders are required for the activation of the compression operation on the clients, the server’s side only requires a single decoder because the incoming client-training-information from all clients is
discontinuous and can be scheduled using the First-In-First-Out rule. This ensures that the hardware requirement of the server can still be satisfied.

Algorithm 1 shows the complete pseudo-code for the HCFL-integrated FL. After the initialization, the server starts its training iterations. Each iteration is named as one communication round. At every communication round, the server uploads global weights to every client and calls for updates from \( K \) selected clients. The selected clients train their models in parallel and then send their new local models to the server. The server decodes all of the received local parameters and then executes the accumulation and averaging to acquire a new global model. This process is continuously processed until the FL achieves the desired convergence or when the max communication round value is reached. Moreover, the encoder compresses the model prior to the packaging and modulating phases of the communication process on the transmitting links. Likewise, the decoder reconstructs the model after the demodulation and de-packaging phases in the receiving links. This closed-loop process ensures the compatibility of HCFL in any IoT systems.

3.3 HCFL Network Architecture

The proposed HCFL framework is expected to compress the model parameter data with a ratio of 1:4 to 1:32 in terms of size. This rate can be achieved by dealing with the loss of data during the encoding scheme. Hence, a deep neural network is leveraged to reduce the loss between the reconstructed and the original data.

3.3.1 Definition of Datasets

To develop HCFL, we first define the dataset for the network. The data are extracted from the model parameter dataset. Instead of extracting the model parameters towards the end of the training, the data prepared for this system is generated after each epoch in each client to assist the compressor in learning the values and spatial distributions of the neural network’s coefficients. The curse of dimensionality and the computation cost at the client’s end were reduced by splitting the data into two components to be trained with two different compressors, namely the convolution kernel dataset and dense network dataset, where elements in each part have the similar dispensational characteristics.

3.3.2 The Proposed Compression System

uses \( V \) fully-connected (FC) layers at the encryptor (Fig. 4a) and \( (l - V) \) FC layers at the extractor (Fig. 4b), where \( l \) is the total hidden layers of the HCFL system to activate the dimensionality reduction. Each FC layer consists of a dense layer followed by a Tanh activation function for each layer node. The usage of Tanh is to guarantee the output of the HCFL in the range \([-1, 1]\), which is the value range of original model parameters. The FC layer also uses an additional batch normalization in the input in order to make the HCFL more

---

**Algorithm 1.** HCFL Compression-Aided FedAvg FL. We Denote \( k \) to be the Client Index in a Total Number of \( K \) Clients; \( E \) is the Local Epochs, \( \eta \) is the Learning Rate, and the Local Mini-Batch Size is Represented as \( B \).

**procedure** `SERVER EXECUTES`

- Initialize \( w_0 \)
- for each round \( t = 1, 2, \ldots \) do
  - \( m \leftarrow \max(1, K \times C) \)
  - \( S_t \leftarrow \) (random set of \( m \) clients)
- for all \( k \in S_t \) in parallel do
  - \( h^k_{t+1} \leftarrow \text{CLIENT UPDATES}(w_t, k) \)
  - \( w^k_{t+1} \leftarrow \text{DECODE}(h^k_{t+1}) \)
  - \( w_{t+1} \leftarrow \frac{1}{k} \sum_{k=1}^{k} w^k_{t+1} \)
- Updates \( w \) to the clients.

**procedure** `CLIENT UPDATES (w, k)`

- \( B \leftarrow \) (divide \( P_t \) into batches of size \( B \))
- for each local epoch \( e \) from 1 to \( E \) do
  - for batch \( b \in B \) do
    - \( w \leftarrow w - \eta \nabla((w; b)) \)
  - \( h = \text{Encode}(w) \) return \( h \) to server
stable and faster by re-centering and re-scaling, as in Fig. 5. The depth of the HCFL hidden layer is set according to the compression ratio of the HCFL and the complexity of the input model. The higher the HCFL compression ratio is, the more FC layers are added into the deep network. As more layers being added to the neural network, the lower bound of the log probability that the model assigns for the training data ascends (which will be presented in Section 5). Thus, the deep neural networks deliver better compression performance than those of shallow or linear neural networks [32], [33].

### 3.3.3 Data Pre-Processing

A detailed description is given below:

- **Data preparation**: The model parameters are stored during the FL execution. To avoid the dataset imbalance, we only fetch the pre-saturated client’s predicting models. Moreover, with the proposed method, the HCFL compressor can learn the model general distribution at every learning state.
- **Data segmentation**: We apply the divide-and-conquer algorithm [34] to break down each individual model parameter into two or more sub-datasets whose distributions have the high mutual information. Therefore, the clustered dataset distributions become simple enough that HCFL can avoid the curse of dimensionality. The performance analysis of the data segmentation technique is proposed in Section 2.

### 3.4 Proposed Training Phase

We adopt the transfer learning technique to implement the HCFL. First, we train a pre-model with a small amount of dataset on the server. By applying augmentation on small dataset on server, the set of model data samples is thus has an additional variation on data distribution [35]. As a result, we incidentally add noise to the gradient, which raises the variance of the norm of the gradient. This high variance thus increases the gradient stochasticity for the pre-train model [36], which improves the model parameters dataset generalization. Therefore, the received dataset is appropriate for the HCFL training phase. Afterward, the model which is obtained at each epoch from the pre-model training phase is utilized for our HCFL model training process. The training process is illustrated in Figs. 4a and 4b. In these figures, the HCFL is trained as demonstrated in (9). As explained in Section 3.1, the trained HCFL model thus has the generalized features of the distributed clients’ models in practice.

### 3.5 Discussion on the Compatibility of HCFL With Asynchronous FL

In its simplest version, our proposed HCFL, like the majority of existing FL approaches, suffers from the stragglers in asynchronous FL. Nevertheless, due to the similar structure with the conventional FL process, various resource allocation techniques (e.g., [37], or [38]) can be integrated into HCFL to mitigate the straggling phenomenon. We reserve further research on this topic for the future.

### 4 PERFORMANCE ANALYSIS ON LOSSY COMPRESSION

In this section, we theoretically analyze the convergence of HCFL-integrated FL when applied in IoT networks which consist of a large number of devices. We first evaluate the performance of FL by considering its impacts on HCFL. We then prove that in an FL process with a lot of clients in general or in FL-integrated massive IoT network system in particular, HCFL can achieve a performance which is as good as that of the conventional FL process. Furthermore, we analyze the performance of HCFL model and provide more detailed analysis on relationship between the model reconstruction error and three features, including model complexity, compression ratio, and original data entropy.

#### 4.1 Convergence Analysis

Using the FedAvg method [5], we can demonstrate the convergence properties of FL when implementing our proposed HCFL scheme. As the number of clients (i.e., IoT devices) increases, it is guaranteed that aggregated weights are close to the initial desired values.

We consider a compression-integrated FL scenario between a single client in a group of selected clients and a server as shown in Fig. 6. The proposed compression consists of two components: the encoder (which is integrated in the clients) and the decoder which extracts the compressed model sent by the client through the propagation channel. By applying the undercomplete autoencoder [2], we can compress the
original data by reducing its dimensions and important information simultaneously. The ratio of input data to the output data of the encoder, which is equal to the ratio of the output data to the input data of the decoder, represents the compression ratio of the model compressor.

The autoencoder only extracts useful information on the distribution of the data due to the reduced dimension of the encoded data. The number of faded features increases as we configure the HCFL at a higher compression ratio [2] (the HCFL compression ratio is defined by the proportion between the encoder input size and the encoder output size). Thus, the decrypted data at the output of the decoder in the server always have a reconstruction error rate in comparison with the original data. The problem of autoencoder error and the capability of the compression in FL is simplified by converting original form of the autoencoder’s representation into another close-form.

**Theorem 1.** Given the model compression distortion rate \( \mathcal{L}(w) \), the uncertainty that the geometrical distance between aggregated model from lossy compression \( w_t \) and ideal aggregated model \( \tilde{w}_t \) is inversely proportional to number of user \( K \).

\[
P(|w_t - \tilde{w}_t| \geq \alpha) \leq \frac{2}{(K\alpha)} \mathcal{L}(w).
\]

**Proof.** The proof is demonstrated in Appendix A, which can be found on the Computer Society Digital Library at http://dx.doi.org/10.1109/TMC.2022.3190501.

The theorem 1 is the proof of convergence of implementation of the aggregations algorithm at the server when the number of clients in the federated network is large enough. Despite the high loss value from the autoencoder, we can achieve the aggregated model, which is close to the ideal value of the model weights without the HCFL compression. For instance, the uncertainty of an event when \( \mathcal{L}(w) = 2.5 \), an expected loss of the aggregated weights compared to the client weights is \( \alpha = 0.01 \), and \( K = 10000 \) FL devices, is: \( P(|w_t - \tilde{w}_t| \geq \alpha) \leq \frac{2}{(10000 \times 0.01)} = 2.5 = 0.0005 \). Therefore, the certainty is equal to 99.95%. According to [4], the number of IoT and mobile devices with MEC capabilities will be extremely large in beyond 5 G and future 6 G systems, i.e., 24 billion devices by 2030 [4]. As a result, the certainty induced by the proposed HCFL system can be further reduced in emerging scenarios with massive IoT connectivities, showing the efficiency and great potential of the HCFL system.

On the one hand, we prove that the distortion rate induced by the HCFL’s lossy compression process is decreased significantly by the FL’s number of participated clients. On the other hand, the distortion rate from the lossy compression also encourages FL models in distributed devices to execute the exploration steps toward gradient descent. This noise acts similarly with the gradient noise, which is proposed in [39].

5 **IMPACT OF MODEL COMPLEXITY TO HCFL SETTINGS**

In the HCFL system, the encoder aims to extract the representative features of the original data. There is a direct mapping through the neural network from each individual particle from the primordial model to the compressed output at the encoder. The compressed data is then mapped straightforwardly to the extracted output at the decoder. The procedure can be described in Fig. 7. As we can see from the figure, there is a representational vector for each pool of original data points that have the same characteristics. The reconstructed vector is sampled from the previously mentioned pool and can represent a group of data points in that pool. Hence, the compressed data is the mutual information between the original and the reconstructed data. The lower the compression ratio is, the more information that the compressed data transfers from the original data to the extractor. Therefore, the fewer the number of original data features that one compressed data component needs to represent, the better the HCFL can perform.

**Theorem 2.** The reconstruction loss \( \mathcal{L}(w) \) of HCFL is proportional to

\[
\mathcal{L}(w) \approx \frac{H(W) - H(C)}{N\log(2\pi e)} - \sum_{i} P(w)\log P(w) - \sum_{c} P(c)\log P(c),
\]

where the distribution of \( C \) follows the joint distribution \( P(X, g^1, \ldots, g^V) \) with \( V \) being the depth of the compressor’s hidden layer. Thus, we have

\[
P(c) = \prod_{i=1}^{V+1} P(g^i | g^{i+1})
\]

\[
= \prod_{i=1}^{V} \prod_{j=1}^{n_i} P(g^i | g^{i+1}) \prod_{k=1}^{M} P(c_k | g^V)
\]

\[
= P(g^1, \ldots, g^V) \prod_{k=1}^{M} P(c_k | g^V),
\]

where \( P(g^1, \ldots, g^V) \) is the joint probability of the \( V \) hidden layers neural network of the compressor and the factorized conditional distribution of the compressed output data.

**Proof.** The proof is demonstrated in Appendix B, available online.
The two Eqs. (11) and (12) give us a prerequisite for the convergence of the HCFL compression. To achieve the HCFL compression system with low restoration error, the HCFL network’s complexity and compression ratio need to be taken into account. When the HCFL with higher compression ratio and more complicated input data is applied, the neural network model is needed to be the deeper. Moreover, the data segmentation technique is considered as a robust method when dealing with models with a huge size and tremendously complex structure.

6 PERFORMANCE EVALUATION

In this part, we evaluate the performance of the proposed HCFL compression. Depending on different testing scenarios, we establish multiple settings to examine the HCFL under distinct conditions in Section 6.1. We then evaluate the HCFL compression efficiency with various compression ratios in Section 6.2. Our experiments show that HCFL can achieve up to 32 times of reduction in data size. Then, we evaluate the impact of clients number on FL convergence rate in Section 6.3. Section 6.4 shows the influence of the client’s predictor hyper-parameters on the HCFL-integrated FL to analyze the suitable setting when applying HCFL on FL.

6.1 Settings

The reconstruction error of the HCFL-compression framework is considered along with the losses of the predictors at the clients’ ends with the updated model parameters under different FL settings to simplify the evaluation. For the simulation, we use a set of 100 clients in synchronous FL system for assessments. A comprehensive description of the setting is presented below.

Evaluation on Communication Efficiency. As mentioned in Section 1, our proposed network model is in compliance with the 3GPP standard. To be more specific, our proposed HCFL is located as the effectiveness encoder in [40, Figure 3a]. Therefore, any package error is pre-processed and corrected via HARQ protocol [41]. Therefore, the encoded data from HCFL is guaranteed to be flawless and independent of the channel model. As a result, we only consider the data compression ratio as a communication efficiency evaluation metric in this research.

Dataset Segmentation. The model parameter datasets of the FL model in our simulations are processed as follows:

- MNIST: The convolutional layers and dense layers are trained in different HCFL compressors and extractors. Each of the HCFL learns the different distribution of each group of convolution kernel parameters or fully connected layers. Hence, we can achieve the high compression efficiency for the HCFL.
- EMNIST: Due to the complexity of the 5-CNN model parameters, we fractionate the dense layers’ parameters into 8 balanced parts in order to reduce the entropy of each part. The HCFL compressor needs to execute eight different trainings and the HCFL compressor setting is stored in the HCFL memory. Whenever the FL model applies the HCFL, the particular HCFL setting is loaded from the memory to process the corresponding segmented dataset.

Initial Implementation Details. Each round, the server selects 10 customers from a pool of 100 clients at random. We will use a fixed learning rate of 0.01 as an example. The number of epochs at each client is set to five, and the minibatch size is set to 64. This implementation is applied to the experiment in Section 6.2. In Sections 6.3 and 6.4, the numbers of participating clients \( K \), local epoch \( E \) and local batch-size \( B \) are applied with different values to evaluate the HCFL under different settings. The detailed settings of those experiments are demonstrated in section 6.3 and 6.4.

6.2 Robustness to Compression Proficiency

We first perform some experiments with the HCFL compression only to validate the affection of the compression ratio on the reconstruction error of the extracted data. We evaluate the algorithm after 100 training rounds with an equal number of training samples for each client, and
every client is computed in 10 independent runs each round for a fair comparison. In this subsection, we consider three main compression efficiency features, including compression efficiency, computational delay, and HCFL-assisted FL accuracy.

From the perspective of compression efficiency, we evaluate the actual compression performance when applying different compression ratio settings in HCFL on FL. Tables 1 and 2 show the reconstruction error and the compression efficiency of HCFL and the benchmarks, including FedAvg [5] and T-FedAvg [22], respectively, under different compression ratio settings. We apply the benchmarks on two datasets: MNIST and EMNIST. Assume that there are 10 users participating in each round over 100 rounds, the total capacity which is necessary for the transmission between clients and the centralized server is 20.5 GB for the LeNet-5 model and 27.2 GB for the 5-CNN model, respectively. This large amount of data is an enormous burden on large-scale IoT networks. The communication cost is expected to be more significant in high complexity deep networks (i.e., AlexNet [45] or ResNet [46]). The relationship between the model parameter size and the communication time is given as

\[ T_{k}^{\text{comm}} = s_k / R_k, \]  

(13)

where \( s_k \) is the model parameter size and \( R_k \) is the transmission rate. From (13), by reducing transmitting model parameter size, we can reduce the communication time with the same ratio. It is worth noticing that when we increase the compression ratio, the communication efficiency is improved. In case HCFL-assisted FL is applied, we can reduce a huge amount of data on the transmission link and the required communication time as well. In particular, there are four proposed compression ratios in our work: 1:4, 1:8, 1:16, and 1:32. The data from the two tables reveals that applying the setting of 1:4 helps to reduce the communication cost to 25.22% and 20.7% when working on LeNet-5 model and 5-CNN. Especially, by applying the 1:32 setting, the communication cost can be reduced to 3.7% (from 20.5 GB in the conventional FL to 757 MB in HCFL-assisted FL) in LeNet-5-integrated clients and 3.35% (from 27.2 GB in the conventional FL to 910.55 MB in HCFL-assisted FL) in 5-CNN-integrated clients. Furthermore, it can be observed from Table 1 that HCFL at compression ratio of 1:32 outperforms the T-FedAvg [22] in communication efficiency as the T-FedAvg’s maximum compression ratio is capped at 16 times.

From the perspective of computational delay, we assess the average computational time for both clients and server in a long-term FL process. Because the HCFL components are deployed in both server and clients, we conduct the delay calculations on both centralized server and distributed clients. We calculate the average delay value over 100 communication rounds in order to get the fair assessment. The computation delay is demonstrated in Table 3. As observed in the table, the assessment is implemented on both LeNet-5-integrated client (MNIST dataset) and 5-CNN-integrated client (EMNIST dataset). The HCFL process time can be calculated as follows:

\[ T_{r}^{\text{HCFL}} = T_{r}^{\text{comp}} - T_{r}^{\text{comp}}^1, \]  

(14)

where \( r \) is the compression ratio of the HCFL system, \( T_{r}^{\text{HCFL}} \) is the HCFL process computation delay and \( T_{r}^{\text{comp}} \) is the client’s total computation delay with the compression ratio \( r \). From the table, we can see that the HCFL process on both clients and server is low (less than 40 milliseconds on client and 350 milliseconds on server). This time amount is much lower than the predicting process delay on the client (around 2.1 to 2.2 seconds).

In the following evaluation of HCFL compression efficiency, we consider the HCFL-assisted FL accuracy. The assessment uses the same setting with ten clients randomly selected from 100 clients in total, the selected client ratio is set at 0.1, and the batch size is set to the maximum possible

---

**TABLE 1**

Performance of HCFL and Other Compression Methods on LeNet-5 Model Training With MNIST Dataset With Different Compression Ratio (Communication Cost in 100 Rounds on IID Data)

| Compress Method | Reconstruction error | Encoded Size (MB) Ratio | True Compress Ratio |
|-----------------|----------------------|-------------------------|--------------------|
| FedAvg          | 0.0                  | 20500/20500             | 1.00               |
| T-FedAvg        | N/A                  | 1281/1281               | 15.999             |
| HCFL 1:4        | 0.0016               | 5170/5170               | 3.965              |
| HCFL 1:8        | 0.0037               | 2620/2620               | 7.824              |
| HCFL 1:16       | 0.0037               | 1370/1370               | 14.963             |
| HCFL 1:32       | 0.0040               | 757/757                 | 27.080             |

Each round, ten out of 100 clients are participated in the training \((C = 0.1)\).

---

**TABLE 2**

Performance of HCFL Compression on 5-CNN Model Training With EMNIST Dataset With Different Compressing Ratio (Communication Cost in 100 Rounds on IID Data)

| Compress Method | Reconstruction error | Encoded Size (MB) Ratio | True Compress Ratio |
|-----------------|----------------------|-------------------------|--------------------|
| FedAvg          | 0.0                  | 27200/27200             | 1.00               |
| T-FedAvg        | N/A                  | 1,650/1,650             | 16.485             |
| HCFL 1:4        | 0.0486               | 5630/5630               | 4.831              |
| HCFL 1:8        | 0.0495               | 2930/2930               | 9.283              |
| HCFL 1:16       | 0.0501               | 1570/1570               | 17.324             |
| HCFL 1:32       | 0.0693               | 910.55/910.55           | 29.872             |

Each round, ten out of 100 clients are participated in the training \((C = 0.1)\).
value (i.e., equal to the data size at each client) with the number of training epoch set at five. To compare the performance between different compression ratios of HCFL, we train different neural network models for the encoder and decoder of each particular HCFL with the determined compression ratio and then embed them into the recommend FL model. Fig. 8 shows the predicting accuracy at clients with different compression ratio settings at the HCFL compressor. The test accuracy is calculated by the percentage of data in the test set with the predicting label matching their original label. As we can see from the figure, the accuracy of HCFL at the beginning of the distributed training session is relatively low due to the internal error of the data generated by the HCFL compressor when working on MNIST dataset with LeNet-5 as the predicting model. However, the performance of the HCFL-integrated FL can be converged after only six to seven communication rounds. Moreover, although the global accuracy is decreased at the compression rate of 16 and 32, the test accuracy at 98% for 1:32 compression rate is at the acceptable threshold. Alternatively, this decline may be attributed to the fact that the more representation of the data vanishes from the original information, the more error-prone that the HCFL-assisted IoT system is exposed to.

When dealing with such model with higher complexity as 5-CNN for the EMNIST, we need to apply the pre-mentioned dataset segmentation technique for the model parameters. The performance of HCFL on the proposed model is demonstrated in Fig. 9. As observed from the figure, due to the random shuffle and division of the full EMNIST dataset for the clients and the stochastic initiation of the client predicting model, the test accuracy of all five cases shows the high fluctuation and different convergence speeds at the primitive stage of the HCFL training. However, the HCFL system can converge within after less than 100 communication rounds, regardless the proposed compression ratio. The high performance of different compression schemes in the prolonged term is believed to attain on account of the reduction in entropy of the model parameters which is proposed in Section 5.

6.3 Contribution of Participating Client Quantity to Global Convergence

In this assessment, we evaluate the impact of the number of participating clients on the convergence of the error-prone compressed data. The HCFL compression is applied in FL with a dissimilar quantity of clients. We evaluate them on two datasets, i.e., MNIST and EMNIST. The results obtained by the HCFL-assisted FL on MNIST and EMNIST are shown in Fig. 10a and Fig. 10b, respectively. In general, it is shown that the performance of the HCFL compression function in the FL model can attain expected efficiency with a various number of clients participating in the FL model. However, the larger the number of clients is, the sooner the system will converge to the global extrema. Therefore, with a large number of clients using HCFL, predictors accuracy can quickly achieve a high performance, and thus the accuracy will be more stable through communication rounds. For example, for the setting \( K = 100 \), the test accuracy of the LeNet-5 model on MNIST dataset reaches 99% after less than 20 communication rounds with a low standard deviation (less than 1%). In contrast, the system with a low number of assigned clients \( (K = 10) \) has a relatively high standard deviation of accuracy (more than 3%) after 80 communication rounds.

In particular, different deep network models being used on the clients make distinct impacts on the whole HCFL-assisted FL. For instance, the complex model 5-CNN with the EMNIST dataset shows a more significant outcome than the mentioned evaluation on the MNIST dataset. For the system with 100 clients in comparison with the 10-client system, the test accuracy is higher with a notably lower variance. Generally, in a predetermined number of training rounds, the number of clients affects the FL process accuracy and convergence speed. As more clients participating in the FL process each round, the model’s absolute precision and training speed suffer from fewer adverse effects. Nevertheless, once \( K \) is improved to a particular level, the advancement of the system performance will be less noteworthy, and sometimes it even sets about degrading. When bringing the FL process into practice, we can face difficulties that as \( K \) expands, more and more clients update their local parameters to the server. As a consequence, the communication and computation cost of the FL model amplify significantly. Fortunately, as we can see in the simulation results in Fig. 10a and Fig. 10b, there always exists an upper bound of performance saturation when \( K \) increases. This is encouraging because in real-world applications with large scale of clients, we only need to select a set of clients from the network to execute the FL process in each communication round. This procedure save a considerable amount of communication cost for the FL process.
In this part, we investigate the impact of compression reconstruction error on different FL process settings. Two client model’s hyper-parameters that we consider are: Epoch and batch size. From the perspective of epoch analysis, as the number of epochs training on each selected client at each communication round increases, the system can achieve significantly better test accuracy (Fig. 11a) and loss (Fig. 11b) after each round. To be more specific, the system in which the number of epochs is set at 20 can converge after few rounds of communications. Meanwhile, the system with a minimum setting of training epoch gets saturated at the accuracy of around 86% and the loss of 0.4. The number of epochs affects the number of iterations that the model completes the training throughout the entire training dataset. Hence, the predictor can achieve the better performance when we train the model with more epochs. However, when the number of epochs becomes too large, the system can suffer from overfitting and extreme computational burden. In this work, we use five epochs for a balanced trade-off between the system computation and the system performance since the convergence of that setting is close to the capability of the pre-mentioned high epoch setting.

**Fig. 10.** Implementations of the FL process applying HCFL for MNIST and EMNIST handwritten digit dataset. In these simulations, we set up the FL with different numbers of clients. (a) Number of clients affects the aggregation accuracy of the FL process on MNIST dataset. (b) Number of clients affects the aggregation accuracy of the FL process on EMNIST dataset.

**Fig. 11.** Aggregation accuracy on LeNet-5 model with MNIST dataset for different local epochs ($E$). Number of total clients and participated ratio are fixed ($K = 100$, $C = 0.1$).

**Fig. 12.** Aggregation accuracy on LeNet-5 model with MNIST dataset for different batch size ($B$). Number of total clients and participated ratio are fixed ($K = 100$, $C = 0.1$), number of epoch ($E = 5$).
In the following analysis, the batch size is taken into account. As observed from Fig. 12a and Fig. 12b, HCFL-integrated FL can achieve better performance when the small batch size is applied. The test loss reflects the categorical cross-entropy between the original and predicted labels of the data collected on clients. The maximum batch size can hardly attain the expected efficiency with a low-par accuracy and loss after 300 communication rounds (60% and more than one respectively). It can be observed that, the setting of batch size can make a big contribution to the computation load of every client. Reducing the batch size helps the networks train faster and requires less memory with minibatches. Moreover, the weight updates have more variance when applying a smaller batch size than when we use the larger batch size. This noise can act as a regularizing effect in weight updating, making the FL operate with better performance.

We make all our code publicly available for reproduction of results and extension.1

7 CONCLUSION AND FUTURE WORKS

A new compression scheme for FL was developed in this study. In particular, the proposed technique possesses several advantages over the conventional technique, such as high compression proficiency, high accuracy, and viability for the FL process (especially the FedAvg algorithm, wherein it is embedded at the server). For example, when the HCFL-integrated FL was set at the high compression ratio (i.e., 1:32), the global test accuracy of the FL process was reduced by approximately 1%–2% in comparison to the baseline method. It was confirmed that the HCFL can be implemented to compensate for the susceptibility to the error of the autoencoder reconstruction output. Further, the low complexity of the proposed compression scheme makes it highly suitable for the FL process, especially the state-of-the-art machine type communication systems such as unmanned aerial vehicle and smart city applications.

In future work, we aim to improve the HCFL proficiency and reduce the supervised-training dependencies by applying deterministic algorithms on the encoder phase. Thus, HCFL is then only needed to train the decoder to adapt to the encoder’s behavior. Moreover, multi-task learning is a promising method to aid the autoencoder’s performance when implemented on deeper deep models such as ResNet, DenseNet, AlexNet, or VGG-16. Hence, HCFL can further be applied to a wide range of FL concepts.

REFERENCES

[1] F. Hussain, S. A. Hassan, R. Hussain, and E. Hossain, “Machine learning for resource management in cellular and IoT networks: Potentials, current solutions, and open challenges,” *IEEE Commun. Surv. Tut.*, vol. 22, no. 2, pp. 1251–1275, Apr.–Jun. 2020.

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[3] X. Chen, D. W. K. Ng, W. Yu, E. G. Larsson, N. Al-Dhahir, and R. Schober, “Massive access for 5G and beyond,” *IEEE J. Sel. Areas Commun.*, vol. 39, no. 3, pp. 615–637, Mar. 2021.

[4] C. De Alwis et al., “Survey on 6G frontiers: Trends, applications, requirements, technologies and future research,” *IEEE Open J. Commun. Soc.*, vol. 2, pp. 836–886, Feb. 2021.

[5] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Proc. Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[6] D. Nguyen et al., “Federated learning meets blockchain in edge computing: Opportunities and challenges,” *IEEE Internet Things J.*, vol. 8, no. 16, pp. 12806–12825, Aug. 2021.

[7] A. Du, Y. Shen, and L. Tseng, “CarML: Distributed machine learning in vehicular clouds,” in *Proc. 26th Annu. Int. Conf. Mobile Comput. Netw.*, 2020, pp. 1–3.

[8] H. Zhang et al., “OnFL: Improving mobile video telephony via online reinforcement learning,” in *Proc. 26th Annu. Int. Conf. Mobile Comput. Netw.*, 2020, pp. 1–14.

[9] Y. Chen, X. Qin, J. Wang, C. Yu, and W. Gao, “FedHealth: A federated transfer learning framework for wearable healthcare,” *IEEE Intell. Syst.*, vol. 35, no. 4, pp. 83–93, Apr. 2020.

[10] W. Y. B. Lim et al., “Federated learning in mobile edge networks: A comprehensive survey,” *IEEE Commun. Surv. Tut.*, vol. 22, no. 3, pp. 2051–2063, Apr. 2020.

[11] Q.-V. Pham, H. T. Nguyen, Z. Han, and W.-J. Hwang, “Coalitional games for computation offloading in NOMA-enabled multi-access edge computing,” *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 1982–1993, Feb. 2020.

[12] M. Chen, N. Shlezinger, H. V. Poor, Y. C. Eldar, and S. Cui, “Communication-efficient federated learning,” *Proc. Nat. Acad. Sci.*, vol. 118, no. 17, Apr. 2021, Art. no. e2024789118.

[13] C. Niu et al., “Billion-scale federated learning on mobile clients: A submodel design with tuneable privacy,” in *Proc. 26th Annu. Int. Conf. Mobile Comput. Netw.*, 2020, pp. 1–14.

[14] J. Hamer, M. Mohri, and A. T. Suresh, “FedBoost: A communication-efficient algorithm for federated learning,” in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 3973–3983.

[15] K. Bonawitz, F. Salehi, J. Konečný, B. McMahan, and M. Gruteser, “Federated learning with autotuned communication-efficient secure aggregation,” in *Proc. 53rd Asilomar Conf. Signals Syst. Comput.*, 2019, pp. 722–726.

[16] K. Bonawitz et al., “Practical secure aggregation for privacy-preserving machine learning,” in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 1175–1191.

[17] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, “Communication-efficient federated learning and permissioned blockchain for digital twin edge networks,” *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2276–2288, Aug. 2021.

[18] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, “Communication-efficient federated learning for digital twin edge networks in industrial IoT,” *IEEE Trans. Ind. Informat.*, vol. 17, no. 8, pp. 5709–5718, Aug. 2021.

[19] Y. Jiang et al., “Model pruning enables efficient federated learning on edge devices,” *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Apr. 25, 2022, doi: 10.1109/TNNLS.2022.3166101.

[20] M. M. Amiri and D. Grouffe, “Federated learning with autotuned communication-efficient secure aggregation,” in *Proc. 53rd Asilomar Conf. Signals Syst. Comput.*, 2019, pp. 722–726.

[21] J. Mills, J. Hu, and G. Min, “Communication-efficient federated learning for wireless edge intelligence in IoT,” *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5986–5994, Jul. 2020.

[22] J. Xu, W. Du, Y. Jin, W. He, and R. Cheng, “Ternary compression for communication-efficient federated learning,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 3, pp. 1162–1176, Dec. 2022.

[23] F. Li and B. Liu, “Ternary weight networks,” 2016, arXiv:1610.04711.

[24] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, “XNOR-Net: ImageNet classification using binary convolutional neural networks,” in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 525–542.

[25] C. Zhu, S. Han, H. Mao, and W. J. Dally, “Trained ternary quantization,” 2016, arXiv:1612.01064.

[26] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach*, 6th ed. London, U.K.: Pearson, 2012.

[27] T. Dumas, A. Rémy, and C. Guillemot, “Autoencoder based image compression: Can the learning be quantization independent?” in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2018, pp. 1188–1192.

[28] V.-D. Nguyen, S. K. Sharma, T. X. Vu, S. Chatzinotas, and B. Ottersten, “Efficient federated learning algorithm for resource allocation in wireless IoT networks,” *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3394–3409, Sep. 2021.

1. https://github.com/skydvn/hcfl-compression
Minh-Duong Nguyen received the BS degree in electronics and telecommunications engineering from the Hanoi University of Science and Technology, Vietnam, in 2016. He was a DSP engineer with Viettel R&D Center from 2017 to 2019. He was working in electronic warfare (i.e., electronic warfare systems and GPS/4G jamming systems). He was a senior embedded engineer with Vinsmart, VinGroup, from 2019 to 2020, developing physical layers for the 5G Base Station. He is currently working toward the PhD degree with the department of information convergence engineering, Pusan National University, South Korea. His research interests include reinforcement learning, federated learning, data representation, and semantic communication.

Sang-Min Lee received the BS degree from the Academic Credit Bank System, South Korea, in 2016. He is currently working toward the master’s degree with the department of information convergence engineering, Pusan National University, South Korea. His research interests include artificial intelligence, federated learning, and 6G communication.

Quoc-Viet Pham (Member, IEEE) received the BS degree in electronics and telecommunications engineering from the Hanoi University of Science and Technology, Vietnam, in 2013, and the PhD degree in telecommunications engineering from Inje University, Republic of Korea, in 2017. He has been a research professor with Pusan National University, Republic of Korea, since Feb. 2020. He is specialized in applying convex optimization, game theory, and machine learning to analyze and optimize edge computing and future wireless communications. He has been granted the Korea NRF Funding for outstanding young researchers for the term 2019–2024. He is an editor of Journal of Network and Computer Applications, Scientific Reports (Nature), and Frontiers in Communications and Networks, and a lead guest editor of IEEE Internet of Things Journal. He was also the recipient of the Best PhD Dissertation Award from Inje University in 2017, the Top Reviewer Award from IEEE Transactions on Vehicular Technology in 2020, as the full globe award 2021 from the Ministry of Science and Technology (Vietnam).

Dinh Thai Hoang (Senior Member, IEEE) received the PhD degree in computer science and engineering from the Nanyang Technological University, Singapore, in 2016. He is currently a faculty member with the School of Electrical and Data Engineering, University of Technology Sydney, Australia. His research interests include emerging topics in wireless communications and networking such as machine learning, edge intelligence, cybersecurity, IoT, and Metaverse. He has received several awards including the Australian Research Council and IEEE Technical Community on Scalable Computing Award for Excellence in Scalable Computing (Early Career Researcher). Currently, he is an editor for IEEE Transactions on Wireless Communications, IEEE Transactions on Cognitive Communications and Networking, IEEE Transactions on Vehicular Technology, and associate editor for IEEE Communications Surveys & Tutorials.

Diep N. Nguyen (Senior Member, IEEE) received the ME degree in electrical and computer engineering from the University of California at San Diego (UCSD) and the PhD degree in electrical and computer engineering from Arizona State University (AZ). He is currently a faculty member with the Faculty of Engineering and Information Technology, University of Technology Sydney, Australia. Before joining UTS, he was a DECRA research fellow with Macquarie University and a member of Technical Staff at Arinc Research Corporation, Boston, MA, USA, and ARCON Corporation, Boston, MA, USA, and consulting the Federal Administration of Aviation on turning detection of UAVs and aircraft, and the U.S. Air Force Research Laboratory on anti-jamming. His research interests include computer networking, wireless communications, and machine learning applications, with emphasis on systems’ performance and security/privacy. He received several awards from LG Electronics, UCSD, UA, the U.S. National Science Foundation, and the Australian Research Council. He is currently an editor, associate editor, guest editor for IEEE Transactions on Mobile Computing, IEEE Access, Sensors Journal, IEEE Open Journal of the Communications Society (OJ-COMS), and Scientific Reports (Nature’s).

Won-Joo Hwang (Senior Member, IEEE) received the BS and MS degrees in computer engineering from Pusan National University, Busan, South Korea, in 1998 and 2000, respectively, and the PhD degree in information systems engineering from Osaka University, Osaka, Japan, in 2002. From 2002 to 2019, he was a full professor with the Inje University, Gimhae, South Korea. Currently, he is a full professor with the Biomedical Convergence Engineering Department, Pusan National University. His research interests include optimization theory, game theory, machine learning and data science for wireless communications and networking.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.