An Efficient Method for Model Compression for Federated Learning Scenarios

Renan Morais[1], **Rafael Veiga**[1], **Lucas Bastos**[1],
Denis Rosário[1], **Eduardo Cerqueira**[1]

[1]Federal University of Pará (UFPA) - Belém, PA, Brazil

rafael.teixeira.silva@icen.ufpa.br,

{renan.morais, lucas.bastos}@itec.ufpa.br, {cerqueira, denis}@ufpa.br

Abstract. *Traditional Machine Learning (ML) relies on centralized systems, requiring large amounts of sensitive data. In contrast, edge processing minimizes data transmission by performing computations locally, improving efficiency. In this context, Federated Learning (FL) is a decentralized machine learning approach that allows clients to train models collaboratively while keeping their data private. However, FL faces challenges such as high communication costs due to the large number of model updates between clients and the central server. In this paper, we introduce an efficient method for model compression for FL scenarios called HUFE-FL, which applies quantization to transform model parameters into discrete values, reducing their size, followed by Huffman encoding. This further compresses the model by efficiently encoding frequent symbols with fewer bits. Our main results show that HUFE-FL achieves up to 77.48% reduction in data transmission compared to traditional FL methods, with minimal loss in accuracy.*

1. Introduction

Federated Learning (FL) emerges as a crucial solution to provide privacy-preserving property with reduced communication cost for future Machine Learning (ML) applications, such as healthcare, finance, mobile applications, and others. FL allows large-scale collaborative learning while keeping the raw data distributed across various client devices. In its operation, FL provides a decentralized ML paradigm by distributing the global model to client devices, which train the model based on their local data [Qi et al. 2024]. Afterward, clients share their model parameters with the aggregation server instead of transmitting their raw data, keeping the raw data on the client devices. The aggregation server receives only locally computed updates to aggregate them for an improved global model [Wang et al. 2024].

FL must deal with statistical differences across clients, *i.e.*., nonindependent and identically distributed (non-IID) data, where the data distributions vary significantly between clients [Sousa et al. 2023]. In the FL scenario, client devices might have heterogeneous computing capabilities, ranging from low-end smartphones to edge devices with dedicated GPUs, such as an Nvidia Jetson [Imteaj et al. 2021]. In addition, clients communicate with the aggregation server to transmit models with a large volume of data using different communication technologies (*e.g.*., 4G, 5G, WI-FI, etc.). Hence, in such

a heterogeneous scenario, it is not practical to train the complete ML model for all the clients in the conventional FL.

In the traditional FL, clients train and transmit complete model updates regardless of the computational and communication constraints. However, FL models might have millions of parameters, leading to a large model size. For instance, ResNet can have more than 25 million parameters. Thus, the communication cost of sending local models over the network could be prohibitive [Haddadpour et al. 2021]. In this sense, the communication cost for transmitting the gradient or model is undermined, as the gradient or model size is still too large [Zhu et al. 2023]. Hence, it is essential to consider ways to reduce the model size to be transmitted over the network [Zhang et al. 2024].

Empirical experiments proved the redundancy of a large portion of model parameters that are not crucial for the whole neural network, leading to unnecessary communication consumption [Wu et al. 2022]. In this context, Personalized Federated Learning (pFL) tailors models to client-specific needs to combat client heterogeneity, providing a personalized model for each client [Ye et al. 2023]. After training, each client builds a model that reflects the unique properties of its training data based on layer importance, allowing each model to be adjusted to the unique characteristics of each client [de Souza et al. 2024]. In the pFL context, layer entropy could be used to indicate the layer importance, which measures both potential loss and gains in the model's loss domain, indicating how different the model is from the trained result [Wang et al. 2025].

It is possible to reduce the communication cost by transmitting compressed local gradients or models to the central server by using quantization operator or Huffman compression [Jiang et al. 2022]. For instance, quantifying the model parameters during training plays a crucial role by converting continuous variables into discrete values, reducing model complexity, and facilitating compression. This discretization makes the model more suitable for efficient compression techniques, such as the Huffman algorithm. Specifically, the Huffman algorithm could then be applied to the quantized parameters to minimize the model size further, where Huffman encoding represents more frequent symbols with fewer bits. This further simplifies the model while minimizing the redundancy without compromising critical information. Hence, it is important to combine both quantization operator and Huffman compression based on layer importance in an efficient way to reduce and compress the transmission of redundant parameters without affecting the global accuracy, which is still an open issue.

In this paper, we introduce the Huffman Encoding Federated Learning (HUFE-FL) method to reduce the size of transmitted models in pFL environments while maintaining model performance and improving response time. To this end, HUFE-FL combines both quantization to maintain discrete parameters and the Huffman compression algorithm to compress the quantized model more efficiently. Specifically, HUFE-FL compresses client models by applying quantization to layer parameters in order to transform parameters into discrete values, reducing complexity and enabling model compression. Afterward, the Huffman algorithm is applied to the quantized parameters to minimize the model size further, especially for less important layers. Evaluation results show that HUFE-FL reduces the data transmission by up to 77.48% compared to the standard FL algorithm while keeping similar accuracy performance.

The rest of this paper is organized as follows. Section 2 presents an overview of works exploring quantization and compression approaches in FL. Section 3 describes our methodology for server-client connection and the HUFE-FL method. Section 4 explores the simulation model and results related to our method. Finally, Section 5 concludes the paper and introduces future work.

2. Related Work

Khan et al. introduced a deep compression method for over-the-air federated learning (OTA-FL) to handle the limits of computation, energy, and communication in edge devices [Khan et al. 2024]. Their approach combines pruning and quantization-aware training (QAT) to shrink model sizes by up to 80% while keeping accuracy close to uncompressed models. Results show that 30% pruning combined with INT8 quantization (30P/8Q) achieves a compression ratio of up to 87% while maintaining an accuracy of 93.0%. These findings highlight the importance of balancing accuracy and compression for efficient model deployment in resource-constrained scenarios.

Beitollahi *et al.* proposed the Federated Learning with Autoencoder Compression (FLAC) method to reduce the size of local models before sending them [Beitollahi and Lu 2022]. FLAC adjusts the compression level based on how much error the system can handle. While this method requires extra computing power for compressing and decompressing the models, it achieves high compression rates, significantly reducing the amount of data sent while keeping the model's accuracy almost unchanged. This issue makes FLAC useful for systems with limited resources, where saving bandwidth is important.

Wen *et al.* introduced Fed2KD, a communication-efficient framework designed for FL that tackles the challenges of non-IID data and communication bottlenecks in Internet of Things environments [Wen et al. 2023]. The framework leverages two key components: a data augmentation process and a knowledge distillation technique. Fed2KD reduces communication overhead by sharing distilled knowledge between devices and servers, using fewer parameters than traditional model weights or gradients. It addresses non-IID challenges by generating balanced datasets with a conditional variational autoencoder (CVAE) trained in federated settings.

Shlezinger *et al.* proposed a FL method that uses universal quantization, a technique based on strong mathematical principles. Their approach tests different ways to compress models while keeping the average squared error as low as possible [Shlezinger et al. 2020]. In this sense, this method reduces the data sent between devices without losing much accuracy. It is a practical solution for saving communication costs in FL systems.

Yue *et al.* proposed an FL approach that uses predictive quantization to analyze client behavior, focusing on accuracy trends over time and the number of iterations required to achieve a target accuracy [Yue et al. 2022]. This method optimizes the representation of model parameters by reducing redundancy and enhancing compression efficiency. Additionally, the authors incorporated arithmetic coding, which utilizes the frequency of quantized residues to estimate the probability table. That enables further compression of the model updates. This dual approach reduces communication overhead while maintaining model performance. It suits scenarios with bandwidth constraints and varying client capabilities.

Shah and Lau proposed a method based on sparse reconstruction to compress global models, reducing downstream communication costs while maintaining accuracy [Shah and Lau 2021]. This work also introduces mask training and saliency-based pruning to compress client models, targeting upstream efficiency during local training. The proposed method uses mask training and saliency criterion, reaching over 60% accuracy on CIFAR-10 within 100 rounds.

Based on the state-of-the-art analysis, we conclude that there are many issues to consider when defining the methods of compressing models in an FL scenario. This analysis highlights the importance of balancing communication efficiency while providing high accuracy and convergence. In this context, a quantization approach combined with entropy-guided Huffman compression is essential to reduce the compression and precision of the ML model. To the best of our knowledge, none of the existing works provide an efficient combination of both quantization operator and Huffman compression based on layer importance to provide an efficient compression while maintaining accuracy with reduced communication overhead.

3. An Efficient Method for Model Compression under pFL Scenarios

This section presents the HUFE-FL method to optimize communication while preserving model accuracy. In its operation, HUFE-FL considers quantization to reduce weight size, while the Huffman algorithm further compresses parameters. The following sections detail the system model and HUFE-FL's operations.

3.1. Scenario overview

Figure 1 shows the pFL scenario overview to address the challenges through layer compression and quantization. In this scenario, we consider a FL scenario involving n devices $\mathcal{P} = \{p_1, \ldots, p_n\}$, where each device is uniquely identified by an index i within the range of $[1, n]$. The learning round is divided into 3 phases, namely: selection, training, and aggregation. During the selection phase, the central server considers a client selection mechanism to select a subset of devices $\mathcal{C} \subseteq \mathcal{P}$ (aka, clients) to participate in training. The server sends the global model (M_g) to the selected clients (p_i), and the model comprises an ML model with l layers denoted as $M \equiv \{layer_1, layer_2, ..., layer_l\}$, where the weight matrix of its neurons characterizes each layer $layer_j$.

During the training phase, the selected clients $\mathcal{C}$ train local models (M_i) using their datasets (D_i), which consist of a set of features $x_{k,i}$, where $D \in \{D_1, \ldots, \|D_i\|\}$, and each feature is associated with a label $y_{k,i}$, and the total number of data samples collected by a specific device p_i is represented by $|D_i|$. It is important to mention that the dataset D_i significantly varies in size, feature, and label distribution, typically showing non-IID characteristics. During the training phase, each client p_i integrates a compression process model into the FL workflow, such as provided by HUFE-FL model.

In this context, each client p_i computes updates to optimize the global model with parameters θ, aiming to minimize the loss function $L_i(\theta)$ and regularization factor H, as shown in Eq. 1. Specifically, the loss $\ell(x_{k,i}; \theta)$ represents the model's prediction for the input $x_{k,i}$ based on the parameters θ, and ℓ is used to compute the distance or dissimilarity between the prediction and the target.

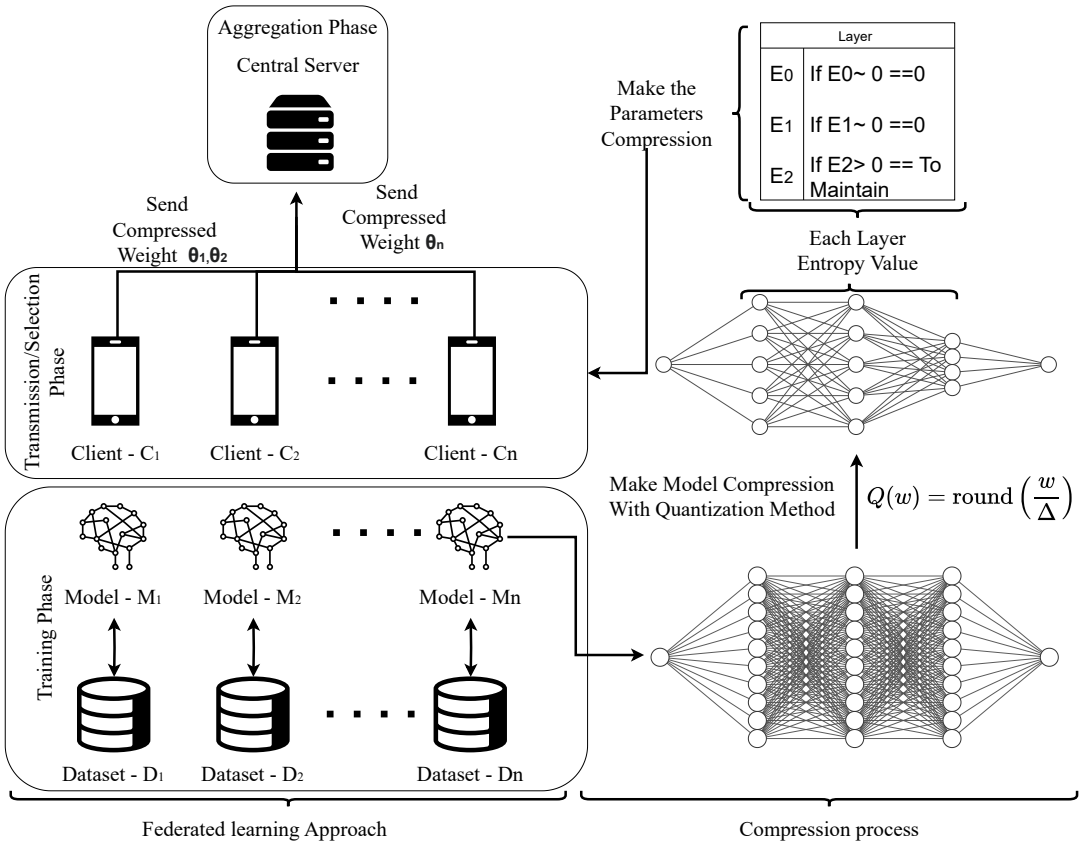


Figure 1. Scenario overview for processing encoder

$$L_i(\theta) = \frac{1}{|D_i|} \sum_{(x_{k,i}, y_{k,i}) \in D_i} \ell(f(x_{k,i}; \theta), y_{k,i}) + \lambda H(\theta), \tag{1}$$

The regularization H factor promotes models with more predictable parameters, contributing to the efficiency of subsequent compression. We consider the Shannon Entropy to quantify the uncertainty or information content of the quantized parameters, capturing how much information is required to encode them efficiently, as represented by Eq. 2. In this way, the $p(x)$ represents the probability distribution of the quantized values, and $\log p(x)$ is the logarithm of the probability. Hence, layer entropy measures both potential loss and gains in the model's loss domain, indicating how different the model is from the trained result to ensure effective and accurate compression [Wang et al. 2025].

$$H = -\sum p(x) \log p(x) \tag{2}$$

Figure 2 shows the relationship between layer entropy values and the model size, directly impacting the encoder and decoder approach with the Huffman approach. In this sense, Huffman encoding becomes more effective when symbols with low entropy—meaning frequent and predictable patterns—represent fewer bits. Reducing entropy enhances compression efficiency by assigning shorter codes to frequently occurring symbols, maximizing representation with minimal bit usage.

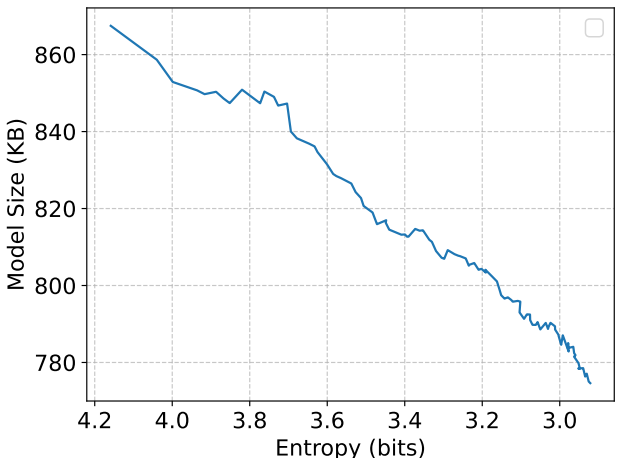


Figure 2. Entropy x model size

We consider the HUFE-FL as the layer compression and quantization into the FL workflow. In this sense, layer entropy E_0, E_1, E_2 could be used to assess each client model's learning state and track changes across training steps, indicating the layer importance [Wang et al. 2025]. In this way, the HUFE-FL model identifies layers with low entropy values (*e.g.*, $E_0 = 0$ or $E_1 > 0$) to prioritize their compression, while layers with negligible entropy ($E_0 = 0$) are discarded. The process involves quantization, where weights (w) are discretized to reduce the precision of parameters, significantly shrinking their size. Afterwards, the model applies Huffman compression to the quantized weights, minimizing data transmission by taking advantage of the reduced entropy in the compressed model. Hence, HUFE-FL model ensures that only the most relevant information is transmitted, significantly reducing the communication load, especially in non-IID scenarios where data distribution varies across clients. The model also balances efficiency and accuracy by compressing model updates and minimizing network overhead.

After completing local training, each client sends their compressed model updated parameters θ_i to the Central Server. At the aggregation phase, the central server aggregate the model updates to refine the global model (M_g) for the next round. In this sense, the central server considers the Federated Averaging (FedAvg) algorithm, where the updated global model parameters w_t at round t are calculated as shown in Eq. 3. The w_t represents the global model parameters, w_t^i denotes the local model parameters from client i, n_i means the size of the local dataset D_i, and $|\mathcal{C}|$ means the number of clients participating in the current round. This weighted averaging approach ensures that clients with larger datasets significantly influence the global model update, effectively addressing data heterogeneity and facilitating convergence. This iterative process continues until the model converges.

$$w_t = \frac{\sum_{i=1}^{|\mathcal{C}|} n_i w_t^i}{\sum_{i=1}^{|\mathcal{C}|} n_i} \tag{3}$$

3.2. HUFE-FL Operations

ML models, such as Convolutional Neural Networks (CNN), contain many parameters that require efficient compression, making the models more compact and easier to store or transmit. In this sense, the HUFE-FL method starts with a layer quantization to reduce the precision of model parameters by limiting the number of unique values used to represent them. In the context of CNNs, quantization is commonly applied to the model's weights and activations, significantly reducing their memory footprint while maintaining the model's performance. Quantization decreases the bit-depth of the data, significantly reducing the overall model size while preserving essential information. The quantization process maps continuous or high-precision values to a smaller set of discrete values. For example, the weights w, originally stored in floating point of 32 bits, are converted to an 8-bit integer representation, as shown in Eq. 4.

$$Q(w) = \text{round}\left(\frac{w}{\Delta}\right) \tag{4}$$

Where Δ represents the Factor scale determined based on the range of values in the model, as shown in Eq. 5. By quantizing the model weights w and activations, we effectively reduce the number of bits b required to store and transmit the model without a significant loss in model accuracy, and also enable faster inference by utilizing lower-precision arithmetic.

$$\Delta = \frac{\max(w) - \min(w)}{2^b - 1} \tag{5}$$

After quantization, Huffman compression algorithm, commonly used for image compression, could be used to reduce the size of quantized parameters. In this sense, Huffman algorithm minimizes the number of bits required to store and send model updates, while leverage the statistical properties of the model's parameters. Specifically, the compressed weights W_{quant} is multiplied by the scaling factor Δ, as shown in Eq. 6. This operation reconstructs an approximate value of the original model parameters, ensuring that the compressed model retains critical information while significantly reducing the storage and transmission overhead.

$$w_{\text{dequant}} = w_{\text{quant}} \cdot \Delta \tag{6}$$

We recalculate the quantized activation functions Z using matrix multiplication with integers, as shown in 7, which makes the calculations simpler and faster. This approach improves efficiency, uses less memory, and speeds up operations, while ensuring they work well with integer-based hardware.

$$Z = A_{\text{quant}} \cdot w_{\text{quant}} \tag{7}$$

Afterwards, Huffman algorithm assigns shorter codes to frequently occurring values, *i.e.*, according to their probability distribution, allowing for a more compact model representation. In this sense, Huffman algorithm is an optimal lossless data compression

algorithm that builds a binary tree, where the most frequent symbols are placed at the root and the less frequent ones are placed further away from the root, as shown in Eq. 8. We used x to represent the symbol, and the prefix is determined by the position of x in the Huffman tree. This property allows Huffman encoding to reach efficient data compression.

$$C(x) = \begin{cases} 0 & \text{if } x \text{ is a left child in the tree} \\ 1 & \text{if } x \text{ is a right child in the tree} \end{cases} \tag{8}$$

For example, Table 1 shows how the frequency of a symbol or in other cases the number can be used by Huffman code. It optimizes model storage and transmission in FL, where communication overhead is a concern. The quantized and entropy-encoded model parameters are smaller, enabling faster server-client communication and improving the efficiency of the compressed model.

Symbol	Frequency	Huffman Code
A	5	011
B	9	010
C	12	100
D	13	101
E	16	11
F	45	0

Table 1. Example Huffman Coding Table

Algorithm 1 sumarizes the HUFE-FL operations to reduce the size of transmitted models in pFL environments while maintaining model performance and improving response time, emphasizing efficient communication between clients and the central server. The algorithm begins by setting up the FL system and distributing the global model M_g to all selected clients. Each client then performs local training on their dataset D_i, generating an updated local model M_i. This local model undergoes quantization, where each parameter w is transformed using the Eq. 4. The quantization step reduces the precision of model weights, minimizing the data size while retaining its critical features. Quantization ensures that the model is ready for further compression without significant loss of accuracy.

Following quantization, each client applies Huffman encoding to the quantized model w_{quant}, producing a highly compressed representation of the local model $M_{\text{compressed},i}$. This compressed model is transmitted to the central server, significantly reducing communication overhead. In the server for each $M_{compressed,i}$ received for our method it needs to bee Decompressed, so in that way for each w_{quant} we must use the $HuffmanDecode(w_{\text{quant}})$ function to add our parameters to an reverse Equation of the compressed models, witch means decompression of the sent models. The compressed models from all clients are aggregated at the server to update the global model M_g. This process ensures that the FL system maintains scalability and efficiency while effectively leveraging the resources of edge devices and minimizing bandwidth usage. The algorithm balances compression efficiency and preserves model performance during training rounds.

Algorithm 1: Huffman Encoder and Decoder for FL

Initialize FL System;
Distribute Global Model M_g to all selected clients;
for *each client* i **do**
    Train Local Model M_i using Local Dataset D_i;
    for *each parameter* w *in model* M_i **do**
        $w_{\text{quant}} = \text{round}(w/\Delta)$;
    end
    $M_{\text{compressed},i} = HuffmanEncode(w_{\text{quant}})$;
    Transmit $M_{\text{compressed},i}$ to the Server;
end
for *each received model* $M_{compressed,i}$ **do**
    $w_{\text{quant}} = HuffmanDecode(w_{\text{quant}})$;
    for *each parameter* w *in model* w_{quant} **do**
        $w_{\text{dequant}} = \text{round}(w * \Delta)$;
    end
end
Aggregate Compressed Models $\{M_{\text{compressed},1}, M_{\text{compressed},2}, \ldots, M_{\text{compressed},n}\}$;
Update Global Model M_g;

4. Evaluation

This section presents the simulation setup employed to evaluate the performance and efficiency of HUFE-FL compared to baseline approaches. We first describe the simulated FL scenario, including the underlying framework, database characteristics, and simulation parameters. We also present the obtained results, focusing on the metrics of accuracy and loss for the global model. Our simulation code can be find on the github [1].

4.1. Simulation Setup

In this paper, we consider a widely used open-source tool called PFLlib[2], which is designed to support various PFL methods. We selected PFLlib because it is widely adopted in the research community and provides flexible, user-friendly features for building and testing FL systems. Its design enables easy customization and personalization of models for each client, which aligns perfectly with the objectives of our paper. Table 2 summarizes the main simulation parameters used in the experiments.

The simulation involves a total of 100 clients, with 30% of them participating in training during each communication round. The experiments consider two datasets, namely, MNIST and CIFAR-100. Each client performs one local epoch per round, ensuring consistent local training updates across all participants. The number of classes varies between 10 (for MNIST) and 100 (for CIFAR-100), reflecting the complexity of the classification tasks. The training runs for 50 rounds on the MNIST dataset and 100 rounds on CIFAR-100, ensuring sufficient iterations for convergence and evaluation of the proposed methods. We evaluate two datasets to determine the accuracy evaluation and bit

[1] HUFE-FL available at `https://github.com/OrenanM/Adaptative-Entropy---FL`
[2] PFLlib available at `https://github.com/TsingZ0/PFLlib`

rate, which will keep their accuracy in more challenging scenarios. CIFAR-100 has more complex and diverse image content and a higher number of classes. CIFAR-100 also makes it harder for models to learn from the data effectively than the simpler grayscale and more homogeneous MNIST images.

The simulation considers two types of data distributions: Dirichlet and Pathological, allowing a comparative analysis of different levels of heterogeneity among clients. In this sense, Dirichlet distribution with a parameter of 0.2 is applied to simulate the non-IID nature of the data distribution across clients, mimicking realistic FL scenarios where client data is inherently imbalanced and diverse. In contrast, a pathological distribution represents an extremely biased scenario with extremely unbalanced data. It challenges the model's robustness by introducing conditions where data from some clients may dominate the training process. This imbalance can lead to model bias and slower convergence.

Table 2. Simulation Parameters

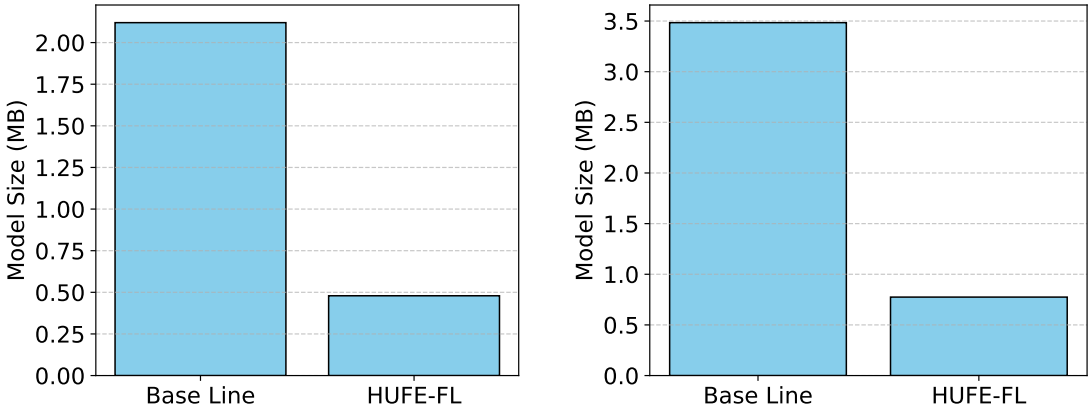
Description	Values
Number of Clients	100 Clients
Training clients	30% of total number of clients
Default dataset	MNIST and CIFAR-100
Local epochs	1 epoch per round
Number of classes	10 and 100 classes
Dirichlet distribution	0.2
Types of distribution	Dirichlet and Pathological
Rounds	50 (MNIST) and 100 (CIFAR-100)

We consider three primary metrics to measure the system's performance, namely, model size, Total Data Transmitted (TDT) until convergence of the model, and the transmitted bytes per client by the accuracy. These metrics allow us to improve communication costs and model maintenance, reaching a similar accuracy without doing any compression.

4.2. Results

Figure 3 shows the model size measurements for the evaluated methods with different datasets. By analyzing the results of Figure 3(a), we can see that HUFE-FL achieves an average size of 0.5 MB, compared to 2.2 MB for the baseline for the MNIST dataset. The model size reduction of 75% is due to HUFE-FL combining both quantization to maintain discrete parameters and the Huffman compression algorithm to compress the quantized model. Similarly, Figure 3 highlights a difference of 2.7 MB between HUFE-FL and the baseline, demonstrating significant compression improvements.

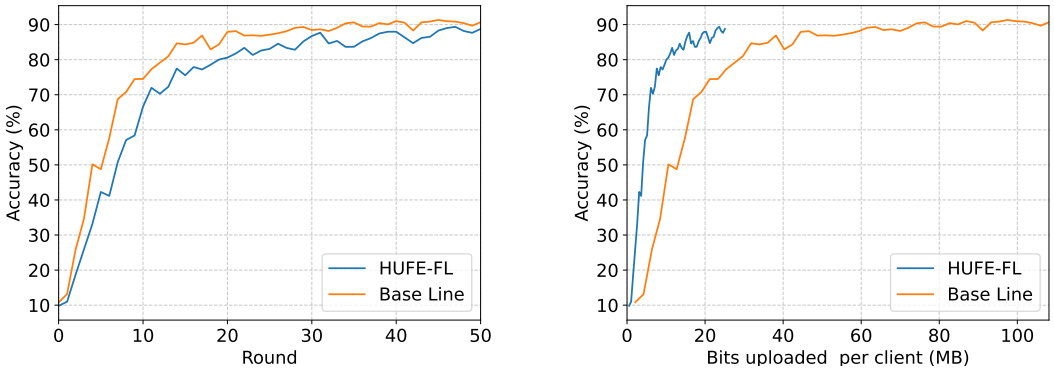
Figure 4 presents the accuracy over rounds and the amount of transmitted bits for the evaluated methods for the MNIST dataset. By analyzing the results of Figure 4(a), we can observe that HUFE-FL achieves similar accuracy to the baseline within 50 rounds, maintaining a difference of less than 10% and surpassing 80% accuracy in just 20 rounds.



(a) MNIST dataset (b) CIFAR-100 dataset

Figure 3. Model size measurements for the evaluated methods with different datasets

On the other hand, Figure 4(b) shows the evaluation of the uploaded bits per client for both base line and HUFE-FL. In the final round, HUFE-FL transmits just over 25MB across all rounds, compared to the baseline, which exceeds 100MB, demonstrating the efficiency of the compressed model. Compression based on entropy in layer parameters helps to maintain accuracy when using the compression results in data that's easier to decode.

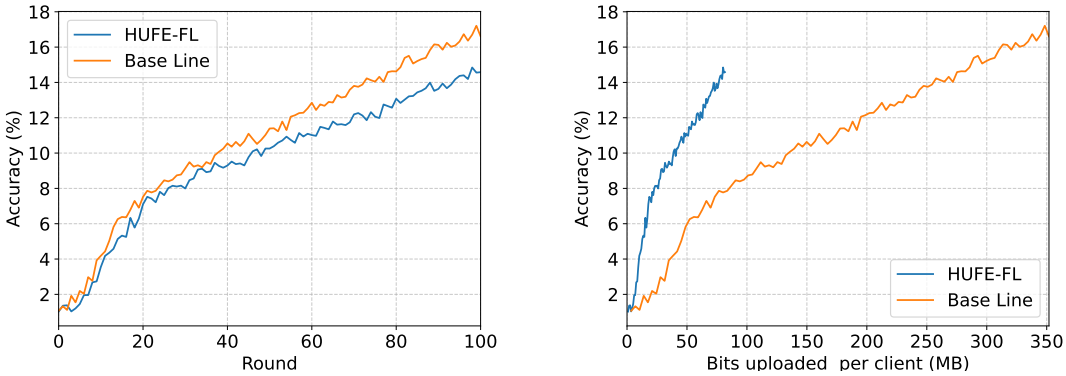


(a) Accuracy over rounds (b) Accuracy over Bits per client

Figure 4. Evaluation results of Accuracy measurements and Bit transferring for MNIST dataset

Figure 5 presents the accuracy over rounds and over amount of transmitted bit for the evaluated methods for the CIFAR-100 dataset. The results of Figure 5(a) revels that HUFE-FL achieves a gradual accuracy improvement, maintaining a similar trend to the baseline. Both methods consistently increase accuracy across 100 rounds, with HUFE-FL reaching similar accuracy compared to base line by the final round. Figure 5(b) shows the bits uploaded per client, demonstrating the impact of compression on network usage. HUFE-FL significantly reduces the amount of data transmitted, requiring only around 125MB per client by the end of the simulation, compared to the baseline, which

exceeds 350MB. This result highlights HUFE-FL's efficiency in reducing communication overhead while maintaining comparable accuracy to the baseline.



(a) Accuracy measurements CIFAR-100 dataset

(b) Bits per client CIFAR-100

Figure 5. Evaluation results of Accuracy measurements and Bit transferring CIFAR-100 dataset

Table 3 shows the communication requirements to train the models until convergence with different datasets and sample distribution settings for both base line and HUFE-FL. We observe that HUFE-FL consistently reduces the communication overhead while maintaining competitive accuracy levels. For the MNIST dataset, HUFE-FL requires only 25 rounds to converge under the non-IID with Pathological (pat) data distribution, transmitting 24.68 MB. In contrast, base line demands 43 rounds and 108.10 MB, resulting in a 77.16% reduction in data transmission. Under the non-IID with Dirichlet (dir) data distribution setting, HUFE-FL further reduces TDT to 25.08 MB. base line remains at 108.10 MB, achieving a 76.79% economy. Similarly, for the CIFAR-100 dataset, HUFE-FL transmits 79.23 MB compared to 351.88 MB of FedAVG, leading to a 77.48% reduction in TDT. Even with these communication savings, HUFE-FL maintains accuracy levels close to base line, particularly in the MNIST dataset, where it reaches 80.5% accuracy in the non-IID (pat) setting and 88.76% in the non-IID (dir) setting. These results highlight HUFE-FL's effectiveness in reducing the communication cost of FL while maintaining comparable model performance.

Table 3. Communication requirements to train the models until convergence in different datasets and sample distribution settings considering the Total Data Transmitted (TDT).

Dataset	Model	Non-IID (pat)			Non-IID (dir)		
		Conv (#)	TDT (MB)	Accuracy (%)	Conv (#)	TDT (MB)	Accuracy (%)
MNIST	HUFE-FL	25	24.68	80.5	18	25.08	88.76
	Base line	43	108.10	77.8	42	108.10	90.68
	Economy (%)	-	77.16	-	-	76.79	-
CIFAR-100	HUFE-FL	100	79.23	16.31	100	81.66	14.58
	Base line	100	351.88	18.60	100	351.88	16.60
	Economy (%)	-	77.48	-	-	76.80	-

5. Conclusion and Discussion

In this paper, we introduced a Huffman Encoding method (HUFE-FL) designed to enhance communication performance in PFL environments, especially in non-IID data and bottlenecks scenarios. HUFE-FL improves the FL model transmission by considering the quantization of weights and Huffman encoding to compress and ensure the quality of the model. Our approach reaches an evaluation of compressing more than 60% of the original model to maintain a constant accuracy. For instance, in the two datasets, the deterioration of the model was evaluated without loss of accuracy and error in their parameters during the encoding compression.

In future work, we plan to extend HUFE-FL by incorporating an additional method with an adaptative local update. For example, clients have their model and do not need to do fine-tuning. We can use its model to aggregate with the sent global model without the need to train to make its own bias of the model. We also aim to investigate the applicability of HUFE-FL in more dynamic FL scenarios, including those with varying quantities of data streaming to each client. FL aggregation strategies, such as adaptive local aggregation, offer an effective solution for enhancing accuracy while compressing the model.

Acknowledgment

This research was partially sponsored by CNPq grant # 404186/2021-1, CAPES, the Brazilian Ministry of Science, Technology, and Innovations, and additional support from Amazon Foundation for Studies and Research Support (FAPESPA) with resources from Law nº 8,248, of October 23, 1991, within the scope of PPI-SOFTEX, coordinated by Softex, and published under Arquitetura Cognitiva (Phase 3), DOU 01245.003479/2024-10.

References

Beitollahi, M. and Lu, N. (2022). Flac: Federated learning with autoencoder compression and convergence guarantee. In *GLOBECOM 2022-2022 IEEE Global Communications Conference*, pages 4589–4594. IEEE.

de Souza, A. M., Maciel, F., da Costa, J. B., Bittencourt, L. F., Cerqueira, E., Loureiro, A. A., and Villas, L. A. (2024). Adaptive client selection with personalization for communication efficient federated learning. *Ad Hoc Networks*, 157:103462.

Haddadpour, F., Kamani, M. M., Mokhtari, A., and Mahdavi, M. (2021). Federated learning with compression: Unified analysis and sharp guarantees. In *International Conference on Artificial Intelligence and Statistics*, pages 2350–2358. PMLR.

Imteaj, A., Thakker, U., Wang, S., Li, J., and Amini, M. H. (2021). A survey on federated learning for resource-constrained iot devices. *IEEE Internet of Things Journal*, 9(1):1–24.

Jiang, Y., Wang, S., Valls, V., Ko, B. J., Lee, W.-H., Leung, K. K., and Tassiulas, L. (2022). Model pruning enables efficient federated learning on edge devices. *IEEE Transactions on Neural Networks and Learning Systems*, 34(12):10374–10386.

Khan, F. M. A., Abou-Zeid, H., and Hassan, S. A. (2024). Deep compression for efficient and accelerated over-the-air federated learning. *IEEE Internet of Things Journal*.

Qi, P., Chiaro, D., Guzzo, A., Ianni, M., Fortino, G., and Piccialli, F. (2024). Model aggregation techniques in federated learning: A comprehensive survey. *Future Generation Computer Systems*, 150:272–293.

Shah, S. M. and Lau, V. K. (2021). Model compression for communication efficient federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, 34(9):5937–5951.

Shlezinger, N., Chen, M., Eldar, Y. C., Poor, H. V., and Cui, S. (2020). Federated learning with quantization constraints. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8851–8855. IEEE.

Sousa, J. L. R., Lobato, W., Rosário, D., Cerqueira, E., and Villas, L. A. (2023). Entropy-based client selection mechanism for vehicular federated environments. In *Proceedings of the 22nd Workshop on Performance of Computer and Communication Systems*, pages 37–48. SBC.

Wang, L., Zhang, X., Su, H., and Zhu, J. (2024). A comprehensive survey of continual learning: theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Wang, X., Jiang, H., Mu, M., and Dong, Y. (2025). A trackable multi-domain collaborative generative adversarial network for rotating machinery fault diagnosis. *Mechanical Systems and Signal Processing*, 224:111950.

Wen, H., Wu, Y., Hu, J., Wang, Z., Duan, H., and Min, G. (2023). Communication-efficient federated learning on non-iid data using two-step knowledge distillation. *IEEE Internet of Things Journal*, 10(19):17307–17322.

Wu, J., Wang, L., and Wang, Y. (2022). An improved cnn-lstm model compression pruning algorithm. In *Advances in Natural Computation, Fuzzy Systems and Knowledge Discovery: Proceedings of the ICNC-FSKD 2021 17*, pages 727–736. Springer.

Ye, M., Fang, X., Du, B., Yuen, P. C., and Tao, D. (2023). Heterogeneous federated learning: State-of-the-art and research challenges. *ACM Computing Surveys*, 56(3):1–44.

Yue, K., Jin, R., Wong, C.-W., and Dai, H. (2022). Communication-efficient federated learning via predictive coding. *IEEE Journal of Selected Topics in Signal Processing*, 16(3):369–380.

Zhang, C., Zhang, W., Wu, Q., Fan, P., Fan, Q., Wang, J., and Letaief, K. B. (2024). Distributed deep reinforcement learning based gradient quantization for federated learning enabled vehicle edge computing. *IEEE Internet of Things Journal*.

Zhu, X., Wang, J., Chen, W., and Sato, K. (2023). Model compression and privacy preserving framework for federated learning. *Future Generation Computer Systems*, 140:376–389.