Available online at www.garmian.edu.krd

Passer Journal of Basic and Applied Sciences

Passer 7 (Issue 1) (2025) 385-398

Original Article

Analysing and Classifying Data Format Strategies for Efficient Communication in Federated Learning

Goran Saman Nariman[1,2], Hozan Khalid Hamarashid[*3]

[1]*Department of Computer Science, College of Science, University of Charmo, Chamchamal 46023, Kurdistan Region, Iraq.*
[2]*Department of Information Technology, College of Science and Technology, University of Human Development, Sulaymaniyah 46001, Kurdistan Region, Iraq.*
[3] *Information Technology Department, Computer Science Institute, Sulaimani Polytechnic University, Sulaymaniyah 46001, Kurdistan Region, Iraq.*

Received 19 October 2024; revised 10 March 2025;
accepted 28 March 2025; available online 10 June 2025

DOI: 10.24271/PSR.2025.484434.1783

ABSTRACT

Federated Learning (FL) enables collaborative model training while preserving data privacy, but it faces significant challenges related to communication overhead and computational efficiency. This paper presents a structured classification of data format strategies used in Federated Learning (FL), dividing them into four main categories: gradients, model parameter weights, knowledge distillation (KD), and federated feature learning (FFL). These strategies are evaluated based on their communication efficiency, computational complexity, and impact on model accuracy. To evaluate the effectiveness of these strategies, the study employed two complementary methods: a comprehensive literature review and an empirical investigation. As the survey confirms, gradient and parameter weight methods are relatively simple, introducing only a small increase in computational cost while providing limited communication reduction. In contrast, KD and FFL offer the most efficient communication strategies but at the expense of higher computational complexity. The empirical study focuses on parameter weight and gradient transmission, as KD and FFL inherently define what is transmitted, making direct comparisons impractical. Experimental results using CNN and LSTM models on MNIST and Shakespeare Corpus datasets reveal that gradient transmission incurs higher computational costs than parameter weights due to additional operations like gradient accumulation and averaging, while the communication overhead remains identical for the same model architecture. The findings highlight the importance of selecting efficient FL communication strategies tailored to specific computational constraints. Furthermore, this study underscores the need for standardized benchmarks to guide the selection of data transmission formats in FL.

Keywords: Federated learning, Communication overhead, Knowledge distillation, Federated feature learning, Transmitting gradients, Transmitting parameter weights.

1 Introduction

Federated Learning (FL) has gained significant attention as a decentralized machine learning paradigm that enables multiple clients to collaboratively train a shared model while keeping their data local. This approach can preserve data privacy and security, as it eliminates the need to transfer sensitive information (such as raw data) to a central server. Furthermore, the total computational cost required to obtain the target model is shared between the central server and the participating clients, for instance, Internet of Things (IoT) devices. Various applications of FL have been identified in the health sector, enabling the sharing of medical data between healthcare providers without compromising patient confidentiality. The Internet of Medical Things (IoMT) is one of the most prominent applications, where FL enhances the integration and analysis of data from diverse medical devices and sensors, aiming to improve machine learning processes that serve patient outcomes and health care services [1,2]. Another important application of FL is in speech recognition systems, which enable collaborative model training without directly exchanging raw data. By ensuring privacy for users, FL can be used to enhance personalization in system customization for various domains [3].

* Corresponding author
E-mail address hozan.khalid@spu.edu.iq (Instructor).
Peer-reviewed under the responsibility of the University of Garmian.

Despite its benefits, FL encounters notable challenges, especially in terms of communication overhead. Frequent model updates between clients and a central server could bring about substantial communication costs, mainly in the presence of a large client pool or complex models. These bottlenecks can significantly limit scalability and efficiency within the FL framework. The challenge is further amplified in resource-constrained environments where bandwidth is scarce, and any improvement in energy efficiency can yield significant benefits. A representative example is mobile and IoT devices involved in FL, which must carefully manage their energy consumption to maintain continuous operation [4].

In FL, the information exchanged between clients and the central server consists of model parameter weights or computed gradients generated during training, rather than raw data. Additionally, several alternative data formats for transmission have been proposed in the literature. These formats aim to capture the most useful features from the clients' learning process and reduce the shared data size through certain techniques, focusing on extracting key features from the dataset while excluding undesirable features.

The efficiency of communication in FL is heavily influenced by the data format used for transmitting model updates. Different data formats can lead to varying levels of communication overhead, computational complexity, and accuracy. For example, transmitting the raw model's parameter weights or gradients may result in high communication costs due to their large size, while more compact representations can reduce the amount of data transmitted but may introduce additional computational effort for encoding and decoding.

While extensive research has been conducted on communication efficiency in FL, there is a notable lack of studies specifically addressing the final data formats transmitted between clients and the server. This review focuses on the most relevant works in communication efficiency to provide context for our study. Existing reviews, such as [5], classify techniques for reducing communication overhead, including model compression, client selection strategies, and optimization algorithms. Similarly, [6] discusses FL's role in privacy preservation and minimizing data transfer but does not focus on specific data formats. To the best of our knowledge, no prior study has systematically classified or evaluated the final data formats used in FL, which is the primary contribution of this work.

In light of these challenges, it is essential to classify and evaluate the various data format strategies used in FL. A study of the trade-offs between communication efficiency, computation cost, and model accuracy can guide the optimization of FL deployments. This paper provides a detailed analysis of a series of data format strategies, which can be categorized into four main types based on the level of data descriptiveness and the levels of detail and abstraction: transmitting gradients, parameter weights, knowledge distillation (KD), and federated feature learning (FFL). The advantages and disadvantages of each approach are discussed to provide detailed insights that can help in choosing the most appropriate data format strategy for specific FL scenarios. Conclusively, this study emphasizes the need for energy-efficient solutions to be considered at the level of client devices to enhance overall sustainability and effectiveness within FL systems.

This paper makes several key contributions to the field of FL. First, it clearly defines what is transmitted between clients and the server, distinguishing between model parameters and other components like architecture or activation functions, which are not shared. Second, it introduces four distinct concepts for knowledge extraction, providing a structured framework for understanding data sharing in FL. Third, it offers a comprehensive comparison of data format strategies, evaluating their impact on computation, communication, and accuracy, serving as a practical guide for selecting the right approach. Finally, it explores the trade-offs between communication efficiency, computational overhead, and accuracy, offering insights to optimize FL systems for diverse application environments.

In addition, Table 1 summarises recent works in the field of communication overhead in FL. The Key Concentration column highlights the main focus of the studies, such as addressing communication constraints, client selection, etc. The next column indicates whether the computational impact was considered in the study, as shown in the table, with all listed works accounting for this aspect. The transmitted data format column shows that none of these studies specifically reviewed the data formats used for transmission, as they focused instead on other elements of communication optimization.

Table 1: Related review studies focused on communication overhead reduction.

Ref, Year	Key concentration	Computational load considered	Reviewed data format for transmission
[7], 2021	Overcoming the Communication Constraints.	Yes	No
[8], 2023	Fundamentals of FL, Communication Deficiency, Client Selection, Resource Management, Optimization Techniques.	Yes	No
[9], 2023	Review and classification of existing communication compression techniques.	Yes	No
[10], 2023	Communication Efficiency, Communication Environment, and Communication Resource Allocation.	Yes	No

2 Material and methods

This section is divided into two main parts: a theoretical analysis based on a literature review and an empirical evaluation relying on practical implementation.

The first part relies on a literature review to establish a classification of the data transmitted between clients and the server. We compare the defined categories by analyzing their properties, such as size reduction, client computational cost, and accuracy, as identified in the literature. Table 2 presents the reviewed papers that address these aspects. The choice of data format for transmitting model updates in FL is pivotal to the success of the training process, as it directly impacts communication overhead, computational efficiency, and model accuracy.

The second part of the study presents an empirical evaluation comparing two widely used strategies in FL: parameter weights and gradients. It assesses their transmission size and computational complexity without applying optimization techniques, offering a baseline understanding of their raw communication costs. This evaluation highlights the fundamental differences between these strategies, particularly in scenarios where no communication efficiency enhancements are implemented.

Table 2: Comparative analysis of FL data format strategies based on communication reduction, additional computation, and accuracy.

Approaches	Size Reduction	Client Computational Cost	Accuracy
Gradient Strategy			
NN Compression [11]	94.5% reduction in total parameters and 18.01× compression ratio.	No contribution to subsequent computation.	0.43% decrease in accuracy.
[12]	Reduction in size of 0.58x.	Energy savings on the edge devices.	Communication reduction comes at a significant cost to accuracy.
[13]	Up to 50x.	Remains manageable and does not significantly increase with the use of 8-bit approximations.	Maintaining high accuracy.
HGC [14]	Compression ratios up to 200x.	Adds computational complexity of $O(n^2)$.	Maintain accuracy close to the baseline model, with only a slight decrease.
QSGD [15]	With AlexNet, it reduces communication time by 4x, and with LSTM, by 6.8x.	With AlexNet, it reduces overall epoch time by 2.5x, and with LSTM, it reduces overall epoch time by 2.7x	Not effected.
MGC [16]	It achieves compression ratios of 880× to 3800× compared to floating gradients.	N/A	Without compromising accuracy.
AdaComp [17]	Reduction of 191× in the total amount of data sent.	Cause additional computation. Highlighting the proposed computation vs communication trade-off.	Improved.
PruneFL [18]	Reducing from 35.88 seconds per round to 1.04 seconds.	Computational time decreased from 11.24 seconds per round to 6.34 seconds as the model density decreased.	Similar accuracy levels.
APF [19]	Reducing 60% without compromising model convergence.	The computation and memory overheads of APF are minimal compared to its benefits in accelerating training.	In some cases, it improves convergence accuracy by reducing overfitting and enhancing the model's ability to generalize.
AdaQuantFL [20]	Much fewer communicated bits compared to fixed quantization level setups	N/A	With minimal or no impact on training and test accuracy.
AQG [21]	The method achieves 18%–50% less transmission compared to popular approaches, including QGD.	N/A	Improved in some cases.
FS-HEAPRIX [22]	Reduction by a factor of 12x.	N/A	Very little loss in test accuracy.

DGC [23]	Achieves a gradient compression ratio from 270× to 600x.	N/A	Not result in any loss of accuracy.
PowerSGD [24]	Two compression regimes (medium and high). At approximately 32 times compression and 128 times high compression.	Added extra computational cost.	Maintaining test performance.
Parameter Weight Strategy			
FedZip [25]	Reduction up to 1085x.	10 to 100 times less battery utilization as packet sizes increase compared to FedAvg and FedSGD.	Insignificant effects on accuracy.
Rank Reduction (RR) [26]	Reduces 13.96× the communication overhead.	Additional computation added.	Minimum accuracy loss.
TFedAVG [27]	Reduction of approximately 34% in communication costs.	The additional computational cost is mainly attributed to the threshold-based verification step that each client has to perform.	Improvements in classification accuracy ranged from 3.01% to 11.09%.
CE-HFD [28]	Reduce up to 191×.	Lighten the computation burden on clients (up to 50% fewer model weights for local training).	It is not significantly affected.
CSRF [29]	Approximately 90% reduction compared to ResNet.	ODENet-50 requires only 65.5% of the computational resources of the original ResNet model. In comparison, dsODENet-50 has an even lower computational cost, just 11.7% of that of ResNet.	Higher accuracy from 12.9% to 11.3%.
iECS-FL [30]	10× compression compared to the baseline method.	O((M + 1)N), higher runtime than the FedAvg algorithm but less than the FedProx algorithm.	Comparable performance to the baseline algorithm.
BiFL-BiML [31]	Clients upload binary parameters instead of 32-bit float point parameters.	N/A	It achieves performance close to conventional FL.
SPARQ-SGD [32]	Around 15K fewer bits than vanilla SGD to achieve the same target accuracy.	It skips local iterations and does not require compression in every round, making it computationally efficient.	It achieves similar performance to state-of-the-art.
T-FedAvg [33]	Reduced by 88% in both the upload and download phases compared to the standard FedAvg.	The computational cost of the proposed method is expected to increase as the depth of the global model increases.	A slight decrease in accuracy is observed as the model depth increases.
FedAT [34]	Reduce communication up to 8.5×, and compression ratio up to 3.5× overall.	Considered.	Increase in prediction accuracy by up to 21.09% compared to existing methods.
ASTW_FedAVG [35]	A notable reduction in communication overhead.	N/A	Overall, the approach has led to improvements in accuracy.
Knowledge Distillation (KD) Strategy			
DS-FL [36]	Reduced up to 99% compared to the FL benchmark, for large image dataset reduced 1.1G parameters to 4 MB, and for text, 228.8 MB to 2MB.	N/A	Improvements range from approximately 20% to 39% across different datasets and tasks.
FD [37]	Around 26× smaller transmission.	N/A	Around 2× higher test accuracy.
CFD [38]	Reduction from 1071.28 MB to 0.101 MB compared to the FedAvg.	Local distillation introduces additional computational overhead.	In some cases, higher compression rates even lead to slight improvements in accuracy.
FedGEMS [39]	Lower communication costs across different accuracy levels.	N/A	It leads to a notable increase in accuracy levels.

FedGKT [40]	It requires 54 to 105 times fewer parameters compared to FedAvg.	It costs 9 to 17 times less computational power (FLOPs).	Maintains model accuracy and achieves slightly higher accuracy in some cases.
FedKD [41]	It can save up to 94.89% of communication costs.	The computational cost is reduced compared to directly learning a large model.	Performance loss of less than 0.1%.
Federated Feature Learning (FFL) Strategy			
CEVFL [42]	Compression rate of 25%, 50%, 75%.	Additional computation for feature extraction and data compression.	It may degrade accuracy as the level of the compression rate.
[43]	Communication reduction by more than 330x.	It is not mentioned in the work, but needs the following extra computation: Infer Probabilities, Generate Pseudo-Labels, and Local Model Updates.	Improves accuracy by more than 45% compared to vanilla VFL.
LESS-VFL [44]	Reaching a target test accuracy while removing at least 80% of the spurious features.	N/A	Does not compromise accuracy.
FedFusion [45]	Reduces communication rounds by more than 60%.	N/A	Increases accuracy and better initialization for new clients.
Two-Stream FL [46]	Reduced by 23.4%.	Adding more computation to each client during optimization.	Improvements in test accuracy.
FedSS [47]	Reduces by 16% per client per round, reaching up to 79% for larger embedding sizes.	Decreased due to fewer parameters being optimized.	Lightly decreased.
JoPEQ [48]	Reduction factor of about 59x.	N/A	Maintaining accuracy in some cases.

2.1 Classifying Data Format Strategies

This subsection classifies four primary data format strategies used in FL, as illustrated in Figure 1. The classification is based on data descriptiveness and the specific techniques for improving communication efficiency. Each strategy involves transmitting different forms of information, which inherently have varying levels of detail and abstraction.

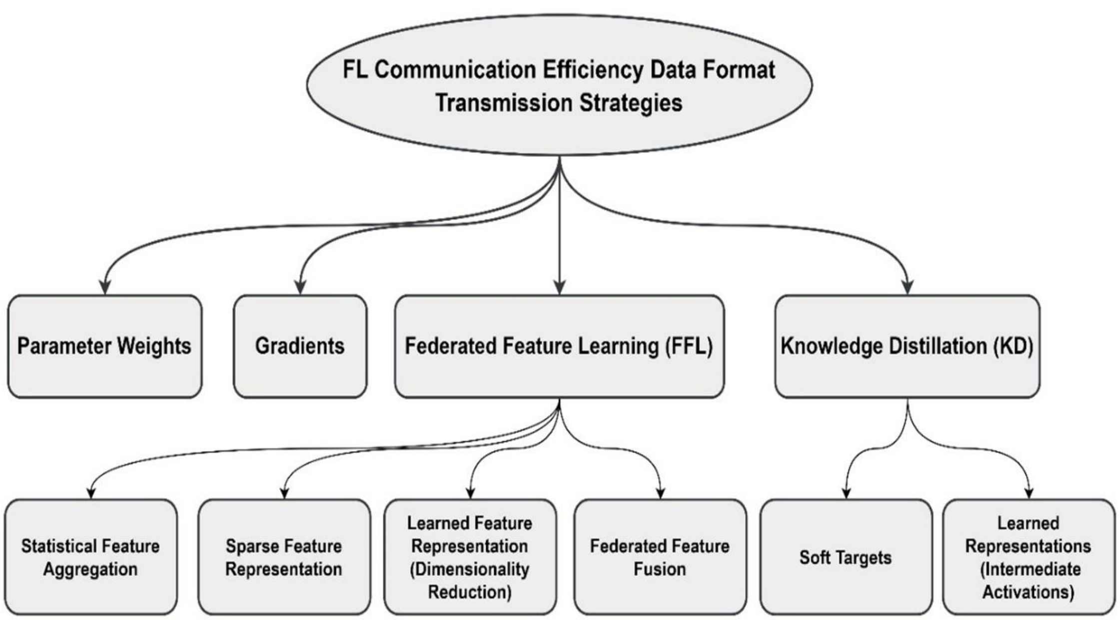

Figure 1: Classification of transmitted data format for communication efficiency in FL.

2. 1. 1 Gradients

The most widely used approach in FL. Each client device independently trains a shared model on its private data. As Figure 2 illustrates, during the local training process, the forward pass is performed first, where the input data passes through the model to produce an output prediction, often via a SoftMax function in classification tasks. The SoftMax function converts the raw output scores (logits) into probabilities that sum to 1, making the predictions interpretable as class probabilities. The loss function then calculates the error between the predicted output and the true labels. During the backward pass (backpropagation), gradients are computed for each model parameter (weights and biases). Specifically, the gradient of the loss concerning each parameter is calculated using the chain rule of calculus. This calculation involves computing the partial derivatives of the loss function with respect to the model's outputs and then propagating these gradients backward through each layer, updating each parameter's gradient accordingly.

For convolutional layers, gradients are computed for the convolutional filters (weights) and biases, whereas for fully connected layers, gradients are calculated for the weight matrices and biases. Once these gradients are computed, they are aggregated into a dense vector format. Each element of these vectors corresponds to a specific parameter in the model, providing a compact representation of the necessary updates.

This data format offers two key advantages: straightforward implementation and direct optimization of the global model. However, while the gradient computation does not add additional computation beyond what is necessary for model training, additional computation arises from the need to serialize, compress, store, and securely transmit the gradients.

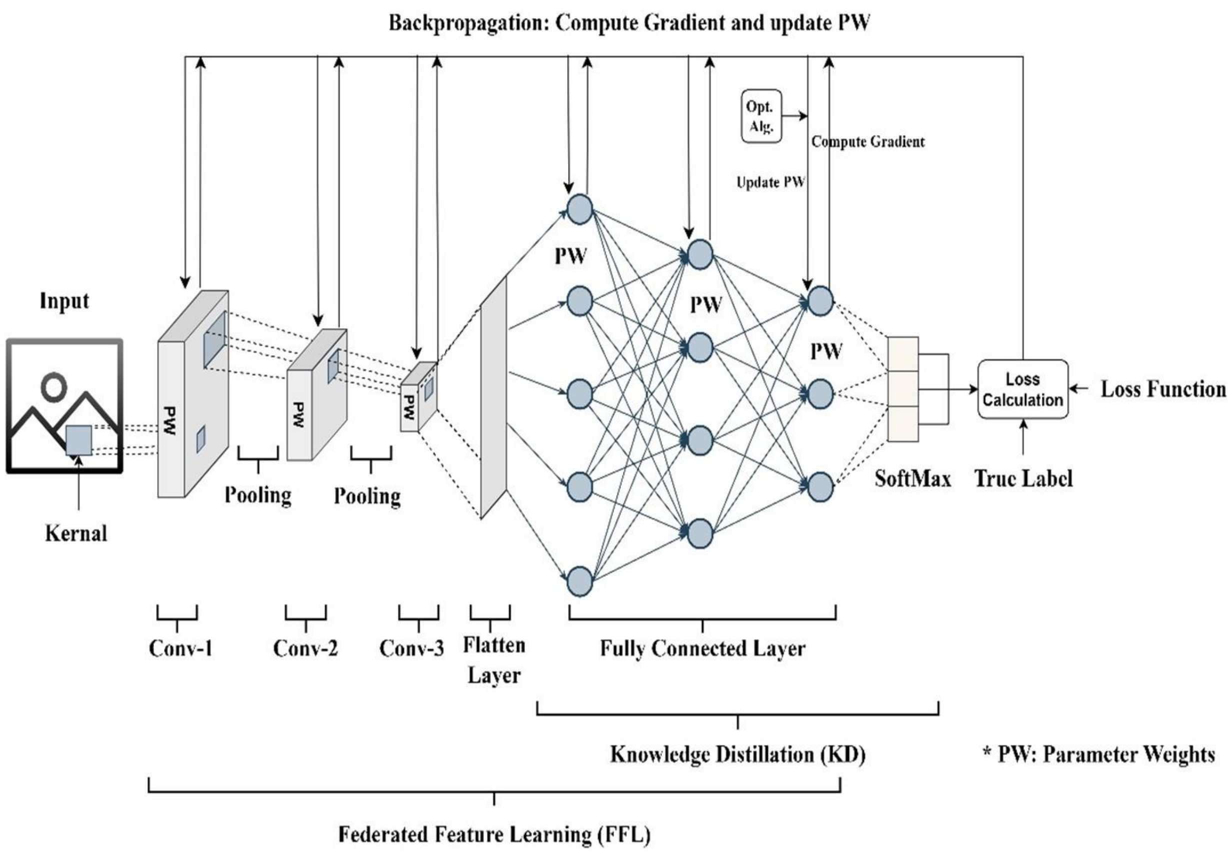

Figure 2: Data Extraction Locations for Transmission in FL Using CNN Architecture.

2. 1. 2 Parameter Weights

This is the simplest approach, where the data format consists solely of the model's parameter weights. In this strategy, clients update their local models iteratively by using the calculated gradient. Once the gradients are computed, each client updates its local model parameters (weights and biases) using an optimization algorithm like gradient descent. The clients then send their complete model weights to the server. The server aggregates the weights (e.g., by averaging) and updates the global model.

Figure 2 depicts the locations of the parameter weights, where the data format is represented as the entire model weights, usually in the form of dense vectors or tensors.

It is worth mentioning that the term Model Scaffolding is recognized in the literature as a technique for enhancing communication efficiency. However, this technique relies on parameter weights as the data format is transmitted between clients and the central server [10].

2. 1. 3 Knowledge Distillation (KD) / Federated Distillation (FD)

KD involves training a smaller, more efficient model (the student) to mimic the behavior of a larger, pre-trained model (the teacher). In a federated setting, the global model typically serves as the teacher, and each client trains a student model using its local data. Instead of transmitting raw gradients or weights, clients send distilled knowledge to the central server. This distilled knowledge can take various forms, depending on the specific KD technique used, and aims to capture the essential information learned by the client models. The format of the knowledge summary can include:

- **Soft Targets:** These refer to the client model's probabilistic predictions (e.g., class probabilities) generated on a held-out validation set. The location is the output layer of the model. Soft targets provide more information than hard labels (i.e., the true class labels) because they convey the model's confidence in its predictions across all classes [36-40]

- **Intermediate Activations:** These are outputs from specific layers in the client's model. The location is the hidden layers of the model and the final hidden layer. Intermediate activations can capture detailed feature representations learned by the model [41].

Figure 1 presents the different possible formats for the knowledge summary, including soft targets (probabilistic outputs) and intermediate activations. Figure 2 shows the locations within the model where KD extraction occurs.

2. 1. 4 Federated Feature Learning (FFL)

FFL focuses on collaboratively learning robust feature representations across clients. Instead of transmitting full model parameters or raw gradients, clients extract feature representations from their local data and transmit these aggregated features to the central server. This approach reduces the amount of data transmitted, lowering communication costs and enhancing privacy. Figure 2 shows the location within the model where these features are extracted. The data format comprises aggregated features, typically represented as vectors or matrices containing statistical summaries or transformed data points. This class can be broken down into four categories, as shown in Figure 1.

- **Statistical Feature Aggregation:** Statistical Feature Aggregation involves calculating statistical summaries directly from the raw data, which are then transmitted to the central server. These summaries, such as means, variances, and histograms, provide aggregated insights without sharing the raw data. Despite being straightforward and efficient in capturing core data characteristics, this approach may overlook finer patterns that more complex feature representations can preserve. These summaries are typically calculated in statistical layers, which are not necessarily part of a neural network but serve to summarize the data [48].

- **Sparse Feature Representation:** Sparse feature representation is an approach where data are represented so that only a small number of features are active (non-zero) at any one time. Such sparsity is achieved through sparse encoding layers, architectural choices, or regularization techniques like L1 regularization. Examples include L1-regularized layers, sparse autoencoders, or special layers for extracting sparse features. Sparse features convey concise information by focusing on the important data aspects and ignoring noise; this can substantially reduce overhead in communication without sacrificing important details [44].

- **Learned Feature Representation (Dimensionality Reduction Techniques):** The learned feature representations of the intermediate layers can abstract information regarding patterns and structures in the data in a meaningful way. Techniques such as Principal Component Analysis (PCA) or t-distributed Stochastic Neighbour Embedding (t-SNE) can also be performed on these outputs to reduce dimensionality while retaining important information. Such features may typically be extracted from convolutional layers in a Convolutional Neural Network (CNN), fully connected layers in a Multi-Layer Perceptron (MLP), or Long Short-Term Memory (LSTM) layers in a recurrent neural network. This represents the trade-off between efficiency and expressiveness, allowing more compact and simultaneously informative feature transmission [42, 43, 47].

- **Federated Feature Fusion:** Federated feature fusion refers to the integration of multi-modal features (e.g., text, images, or sensor data) within a federated learning framework. This is achieved through dedicated network branches, where each branch processes a distinct data modality. The fusion occurs in deeper layers, typically via concatenation of modality-specific features, followed by fully connected layers for unified representation learning. This approach enhances the ability of models to learn complex relationships between diverse kinds of data, resulting in increased depth and accuracy in the model's training [45,46].

2. 2 Method for Empirical Evaluation of Parameter Weights and Gradient Strategies

This section empirically evaluates the two dominant communication strategies in FL: parameter weight transmission and gradient transmission between clients and the central server. This comparison aims to evaluate the baseline computational complexity and transmission data size of these strategies, excluding any communication optimization techniques. This approach reveals the fundamental differences between these two FL communication strategies when used without optimization, particularly in their computational costs and data transmission sizes.

Given the extensive adoption of FL across various domains, the choice of datasets and models for evaluation is based on common practices identified in existing literature. According to prior studies [49], the most frequently used datasets in FL frameworks include the MNIST [50] dataset for image classification tasks and the Shakespeare Corpus [51] for text-based applications. Correspondingly, the most commonly employed models are CNNs for image processing and LSTM networks for text-based learning tasks. These dataset and model selections align with standard FL practices, enabling a meaningful comparison of transmission sizes and computational complexity.

2. 2. 1 Transmission Size and Computation Time of Parameter Weights and Gradients

A critical factor in FL communication efficiency is the size of transmitted data. In standard FL setups, datasets are distributed across multiple clients. Following prior research, such as [52-56], this study assumes data is split among 100 clients, meaning each client handles 1% of the dataset. The experiments are designed around two scenarios: (i) a small-scale setup using 1% of the MNIST and Shakespeare Corpus datasets per client for controlled, computationally manageable evaluation, and (ii) a full-scale setup using the entire dataset (100%) for broad assessments.

The small-scale experiments were repeated 100 times and the full-scale experiments 25 times to ensure reliable results. This repetition minimizes training time fluctuations and provides a more accurate estimate of average computational costs.

Transmission size is measured by the raw data sent from clients to the server during training. Computational complexity is assessed by tracking the time needed for training iterations, including the effort to serialize, store, and transmit the data formats. Figure 3 illustrates the methodology for evaluating the transmission size and computational complexity of model parameters and gradients in FL. The results from these experiments serve as a baseline comparison to understand the raw communication costs associated with these two strategies before applying any optimization techniques.

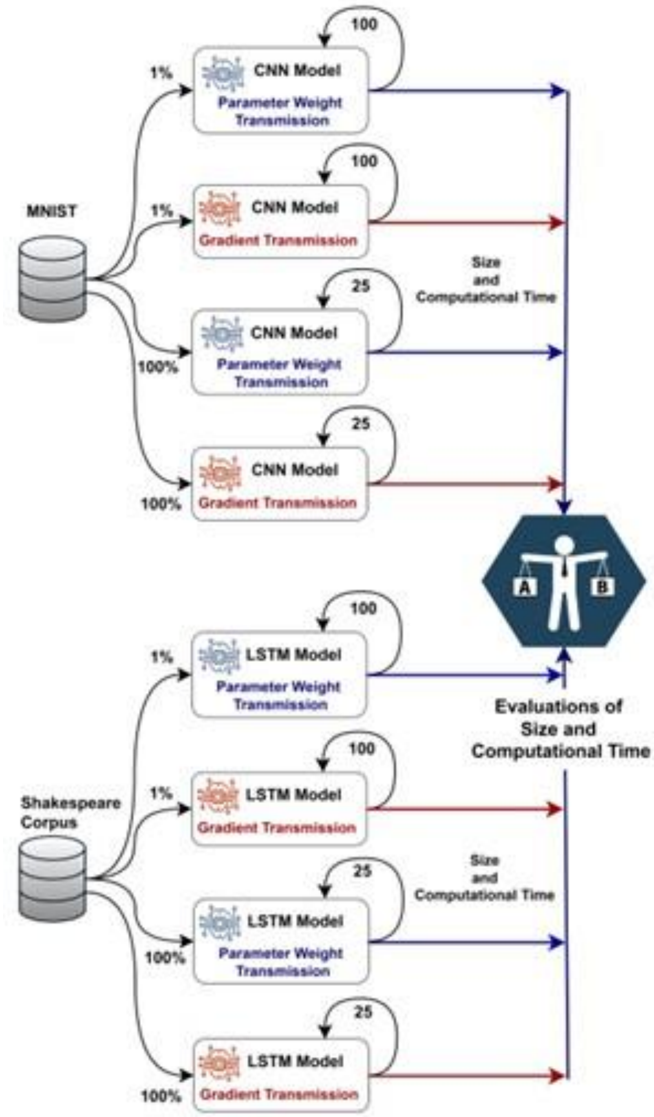

Figure 3: Evaluation method for transmission size and computation time of model parameters and gradients in FL.

2. 2. 2 Justification for Excluding KD and FFL

While parameter weights and gradient transmission are empirically evaluated, KD and FFL are excluded from this comparison. This exclusion stems from fundamental differences in how these strategies handle data transmission. Unlike gradients and parameter weights, which involve direct, granular model updates, KD and FFL transmit abstracted representations such as soft targets, feature activations, or distilled knowledge. These representations vary in size and content, making direct empirical comparison with gradient and parameter weight strategies impractical.

Additionally, KD and FFL rely on advanced techniques like knowledge compression, feature extraction, and task-specific adaptations, which introduce significant complexity. These factors extend beyond the scope of this study, which focuses solely on raw transmission size and computational complexity. While KD and FFL are valuable for FL communication, their evaluation requires a tailored experimental setup to account for their unique properties.

3 Comparative Analysis of FL Data Format Strategies

This section is divided into two parts. The first part provides a comparative analysis of FL data format strategies based on insights from the literature, focusing on communication reduction, computational cost, and accuracy. The second part will present the evaluation of empirical results for parameter weights and gradient strategies, offering a practical perspective on their performance in FL systems.

3. 1 Literature-Based Analysis of FL Data Format Strategies

This section critically evaluates data format strategies in FL based on three key factors: communication reduction, computational overhead, and accuracy preservation. The analysis focuses on the four primary data format classes mentioned earlier. However, due to the significant diversity in the approaches to communication overhead reduction, such as different generated data formats, datasets used, model architectures, comparative models, client numbers, and data distributions, comparing these methods is challenging. For instance, improved convergence speed may reduce the communication rounds; in this case, even if some additional computation is introduced, it cannot be counted as an additional computation cost for the clients because fewer communication rounds have been achieved. Thus, without a strict benchmark, exact comparisons are impossible. Nevertheless, such an analysis is necessary to evaluate data format strategies. To the best of our knowledge, no work has focused on this aspect. This analysis is based on the information provided in Table 2.

3. 1. 1 Communication Reduction

Gradient-based strategies, such as HGC and DGC, aim to reduce communication by compressing gradients. HGC achieves up to 135× compression through gradient quantization and sparsification, while DGC achieves compression ratios ranging from 270× to 600×. These methods are effective but may require additional computational steps for compression and decompression.

Parameter weight strategies also focus on reducing communication overhead by compressing model parameters. Techniques like FedZip and CE-HFD demonstrate significant communication reductions, with FedZip achieving up to 1085× and CE-HFD reducing communication by up to 191x. These methods are generally effective but might introduce some computational overhead due to the need for compression and decompression.

KD methods, such as FD and CFD, are highly effective in reducing communication costs. FD achieves around 26× smaller communication, while CFD reduces the communication size from 1071.28 MB to 0.101 MB compared to FedAvg. These methods are particularly efficient in reducing communication but may introduce additional local computations.

FFL methods aim to reduce communication by learning compact feature representations. The CEVFL technique achieves three distinct compression rates: 25%, 50%, and 75%. These methods effectively reduce communication but might impact the model's accuracy if the compression rate is too high.

Among these approaches, KD and FFL achieve the most significant communication reductions by effectively condensing model updates into compact representations. Gradient-based methods and parameter weights also provide significant reductions by transmitting essential model information, but KD and FFL are particularly advantageous in scenarios where minimizing communication overhead is critical. Overall, the choice of strategy depends on the specific requirements of the FL application, with KD and FFL standing out for their high communication efficiency.

3. 1. 2 Additional Computational Cost

Evaluating additional computation costs is essential for determining the general efficiency of various data format strategies in FL. However, most of these works do not indicate or even estimate additional computational costs for the proposed approaches, making it difficult to draw definitive conclusions from the literature.

KD often introduces more computational steps on the client side, as model distillation processes and compression procedures are widely needed. While these processes reduce communication overhead, they require computationally intensive efforts to transform detailed model parameters into a more abstract and compact representation. Despite this, KD can be efficient when clients have sufficient computational resources to handle these additional steps.

FFL is also prone to additional computational overheads, particularly during feature representation extraction and compression processes. Such features are too expensive to extract from high-dimensional data. However, this additional computational cost is compensated for by the huge reduction in communication overhead once the systems choose FFL, leading to an overall efficient training process in bandwidth-constrained environments.

While gradient-based approaches typically require less computation than KD and FFL, they still demand significant resources for calculating and transmitting gradient updates. The computational and communication overhead varies with model complexity and update frequency. Optimization techniques like gradient sparsification can reduce these demands, but often require precise tuning and thorough evaluation to maintain model performance.

While parameter weight transmission typically requires less computation than KD or FFL approaches, which involve additional compression steps, it can still incur significant computational overhead for large models or frequent updates. Quantization and model pruning can reduce such costs, but these methods may degrade model accuracy.

In summary, while KD and FFL generally offer better communication efficiency, they are usually traded with increasing computation costs. Gradient-based methods and parameter weight transmission typically incur lower computation costs than other approaches, though their communication efficiency depends on implementation specifics. Most of the literature does not discuss the overall assessment of the computation costs in detail. These findings necessitate further investigation to fully characterize FL trade-offs and improve strategy adaptation.

3. 1. 3 Accuracy

Accuracy preservation constitutes a core consideration for FL data strategies. Different methods vary substantially in their effects on global model performance, necessitating careful comparative evaluation.

KD has the potential to enhance model accuracy by utilizing a teacher-student framework, where the student model learns from the predictions of the teacher model. This approach achieves dual objectives: model compression while preserving critical learned features, thereby maintaining (and potentially improving) predictive accuracy. However, the effectiveness of KD is highly dependent on the quality of the teacher model and the distillation process. If these elements are not carefully managed, the outcome may be less than optimal.

As previously noted, FFL emphasizes learning and sharing feature representations rather than entire model parameters or gradients. By concentrating on the most relevant features and filtering out noise and irrelevant information, this approach can enhance privacy while maintaining, or even improving, model accuracy. FFL may be effective, especially in scenarios with non-IID data, where feature extraction may help in further improving the robustness and accuracy of models.

While gradient-based methods are relatively simple to implement, they often struggle to maintain accuracy in heterogeneous data environments due to the risk of gradient divergence. Techniques such as gradient clipping, normalization, and aggregation strategies can help mitigate these issues, but they require careful tuning to ensure they do not negatively impact model performance.

Direct transmission of model parameter weights can help preserve high accuracy, as it involves sharing the full model structure and its learned parameters. However, this approach can present challenges in terms of scalability and communication efficiency, particularly with large models. Although frequent updates and synchronization are essential for maintaining performance, they can quickly become computationally and communication-intensive.

3. 1. 4 Overall Analysis

From the standpoint of reducing communication overhead, both KD and FFL are notable for their ability to significantly minimize the amount of data transmitted between clients and the central server. They compress essential information by focusing on distilled knowledge or feature representations. Gradient-based methods and parameter weights also offer communication savings but to a lesser extent.

Considering the computation cost, all the methods introduce some computation overhead: KD introduces the distillation process, while FFL needs feature extraction and representation learning. Gradient-based and parameter weight-sharing methods tend to be lightweight on the client side; however, they require efficient and effective aggregation strategies on the server side to maintain overall model performance. Much of the existing literature overlooks the additional computational overhead introduced by various learning strategies, highlighting an important opportunity for future research to explore and address these hidden costs.

Accuracy preservation is crucial. KD and FFL can enhance or maintain accuracy through effective KD and feature focus. Gradient-based methods and parameter weights, while their settings are similar in approach, suffer from problems regarding data heterogeneity that affect global model accuracy.

Ultimately, selecting a data format strategy requires careful trade-offs between communication efficiency, computational overhead, and model accuracy. KD and FFL are particularly advantageous for reducing communication overhead while maintaining accuracy. Gradient-based methods and parameter weights are effective but require robust mechanisms to manage the communication overhead challenges without negatively affecting performance and computation complexity. Establishing comprehensive benchmarks for these strategies under standardized conditions is essential for future research, guiding the development of more efficient FL systems.

3.2 Experimental Evaluation of Parameter Weights and Gradient Strategies

This section reports the results of an empirical evaluation comparing the computational time and transmission size associated with parameter weight-sharing and gradient-based strategies in FL. The evaluation considers two scenarios: a small-scale setup using 1% of the dataset and a large-scale setup using the full dataset, with results averaged over 100 and 25 runs, respectively. The main goal is to assess the computational efficiency and communication overhead of both approaches in their raw form, without the influence of any optimization techniques. Accuracy is not considered in this evaluation, as both parameter weights and gradients carry exactly the same information, ensuring no impact on the aggregated model's performance. All experiments were conducted on a high-performance computing system with an Intel Xeon E5-2650 v2 CPU (14 cores, 28 threads), 96 GB RAM, running Windows Server 2019. The implementation was carried out using PyTorch.

Table 3 summarizes the experimental findings for CNNs and LSTM models. The CNN architecture consists of two convolutional layers (32 and 64 filters, 3×3 kernel, ReLU activation, and max-pooling) followed by a fully connected layer with 128 neurons. The LSTM model includes an embedding layer, two LSTM layers (256 hidden units), and a fully connected output layer. These architectures reflect standard choices for image and sequential data tasks in FL.

Table 3: Experimental results of parameter weights and gradients for CNN and LSTM: computational time in second and size in KB for 1% (avg of 100 runs) and 100% (avg of 25 runs) datasets.

Approaches and Datasets	Strategies	1% of datasets, Avg. of 100 runs.		100% of datasets, Avg. of 25 runs.	
		Computational Time (Second)	Size (KB)	Computational Time (Second)	Size (KB)
CNN with MNIST	Parameters	3.04	1647.04	391.85	1647.04
	Gradient	4.14	1647.04	415.43	1647.04
LSTM with Shakespeare Corpus	Parameters	46.03	3180.04	5947.54	3180.04
	Gradient	65.64	3180.04	7135.62	3180.04

Under the 1% dataset configuration, where each experiment was run 100 times to ensure consistency, the results demonstrate clear and consistent differences in computational performance between the evaluated strategies. In CNNs, transmitting parameter weights took 3.04 seconds. In comparison, gradients required 4.14 seconds, a 26.8% increase due to the additional computation involved in accumulating gradients across all batches, averaging them before transmission, and handling gradient storage. These steps introduce extra operations compared to parameter weight transmission, where only the final trained weights are sent. However, this increase remains minor, indicating that gradient-based updates do not impose a significant computational burden at this scale. In contrast, LSTM models require significantly more time due to their sequential nature. Parameter weight transmission took 46.03 seconds, whereas gradient transmission required 65.64 seconds, a marginal 29.9% increase. The findings suggest that, in the case of LSTM models, gradient computations are a major contributor to overall computational demand, amplifying the already high processing requirements due to their sequential nature.

When testing with complete datasets (25 experimental runs), parameter weight transmission demonstrated significantly different computational requirements compared to gradient transmission. In CNN models, transmitting parameter weights took 391.85 seconds, while gradient transmission required 415.43 seconds, reflecting a 5.7% increase due to the additional computational overhead of accumulating, averaging, and storing gradients before transmission. In contrast, for LSTM models, parameter weight transmission took 5947.54 seconds, whereas gradient transmission required 7135.62 seconds, resulting in a 16.65% increase. These findings demonstrate that while CNNs exhibit only moderate computational increases with gradient transmission, LSTMs incur significantly higher costs due to their sequential processing architecture. The additional overhead of gradient accumulation and storage becomes more pronounced in LSTMs, making gradient-based updates notably more expensive than parameter weight transmission for such models.

Regarding data transmission size, both strategies resulted in identical transmission sizes for both dataset size scenarios. For CNN, both parameter weights and gradients required 1647.04 KB, while for LSTM, the size was 3180.04 KB. These results indicate that transmission size depends principally on model architecture, not on the transmission strategy or dataset scale. The key factors affecting the size of both parameter weights and gradients include the number of layers, the number of trainable parameters per layer, Kernel size and filter count (for CNNs), and the specific connections between neurons. Larger models with deeper architectures or more complex layers naturally require larger storage and transmission capacity. Despite these variations, the inherent structure of neural networks ensures that parameter weights and gradients remain comparable in size. These results demonstrate that compression or sparsification techniques remain essential, as raw gradient transmission alone cannot substantially reduce communication overhead.

These results reinforce that gradient-based transmission introduces computational overhead but does not inherently enhance communication efficiency unless combined with additional optimizations

Conclusion

This study provides a comprehensive classification and evaluation of data format strategies in FL, focusing on their impact on communication efficiency, computational cost, and model accuracy. The analysis classifies FL communication strategies into four primary categories: gradients, parameter weights, knowledge distillation (KD), and federated feature learning (FFL), each offering unique trade-offs regarding efficiency, complexity, and performance. To ensure rigorous evaluation, we combined two complementary approaches: a systematic literature-based analysis and a controlled empirical study. The literature-based analysis highlights that while parameter weight and gradient transmission remain the most widely adopted approaches due to their simplicity, they provide limited communication reduction. Conversely, KD and FFL achieve superior communication efficiency but introduce significant computational overhead, making them less suitable for resource-constrained environments. The empirical evaluation further examines parameter weight and gradient transmission using CNN and LSTM models on MNIST and Shakespeare Corpus datasets. The results confirm that gradient transmission demands higher computational resources than parameter weights due to the additional steps of gradient accumulation, storing and averaging. However, despite this increased computation time, both strategies have the same communication overhead for a given model architecture. The findings also highlight that LSTMs exhibit significantly higher computational demands than CNNs, underscoring the challenges of applying sequence-based models in FL. Future research should examine how compression techniques differentially impact parameter weight and gradient transmission, determining which approach exhibits greater compression vulnerability while better maintaining model accuracy and convergence stability.

Authors contribution

Goran contributed to the conceptualization, methodology, literature review, formal analysis, software, validation, writing of the original draft, and the revision (including language refinement). Dr. Hozan provided supervision, validation, and contributed to writing, and editing.

Conflict of interests:

The authors declare no competing interests.

Acknowledgements:

We sincerely thank the University of Human Development, Kurdistan Region, Iraq, for providing access to a high-performance computer server, which facilitating the implementation and testing of the proposed evaluation approaches.

References

[1] Nariman, G. S. & Hamarashid, H. K. Hierarchical federated learning for health trend prediction and anomaly detection using pharmacy data: from zone to national scale. *Int J Data Sci Anal*, 756-5 doi:10.1007/s41060-025 (2025).

[2] Alamleh, A., Albahri, O. S., Zaidan, A. A., Albahri, A. S., Alamoodi, A. H., Zaidan, B. B., Qahtan, S., Alsatar, H. A., Al-Samarraay, M. S. & Jasim, A. N. Federated learning for IoMT applications: A standardization and benchmarking framework of intrusion detection systems. *IEEE J Biomed Heal Info* **27**, 878-887, doi:10.1109/JBHI.2022.3167256 (2023).

[3] Chen, Z. & Xu, S. Learning domain-heterogeneous speaker recognition systems with personalized continual federated learning. *EURASIP J Audio Speech Music Process* **33**, 15-25 (2023).

[4] da Silva, L. G. F., Sadok, D. F. H. & Endo, P. T. Resource optimizing federated learning for use with IoT: A systematic review. *J Parallel Distrib Comput* **175**, 92-108, doi:10.1016/j.jpdc.2023.01.006 (2023).

[5] Wen, J., Zhang, Z., Lan, Y., Cui, Z., Cai, J. & Zhang, W. A survey on federated learning: challenges and applications. *Int J Mach Learn Cyber* **14**, 513-535, doi:10.1007/s13042-022-01647-y (2023).

[6] Almanifi, O. R. A., Chow, C.-O., Tham, M.-L., Chuah, J. H. & Kanesan, J. Communication and computation efficiency in Federated Learning: A survey. *Internet Things* **22**, 100742, doi:10.1016/j.iot.2023.100742 (2023).

[7] Shahid, O., Pouriyeh, S., Parizi, R. M., Sheng, Q. Z., Srivastava, G. & Zhao, L. Communication efficiency in federated learning: Achievements and challenges. *arXiv preprint arXiv* **2107**.10996 (2021).

[8] Asad, M., Shaukat, S., Hu, D., Wang, Z., Javanmardi, E., Nakazato, J. & Tsukada, M. Limitations and future aspects of communication costs in federated learning: A survey. *Sensors* **23**, 7358, doi:10.3390/s23177358 (2023).

[9] Wang, Z., Wen, M., Xu, Y., Zhou, Y., Wang, J. H. & Zhang, L. Communication compression techniques in distributed deep learning: A survey. *J Syst Archit* **142**, 102927, doi:10.1016/j.sysarc.2023.102927 (2023).

[10] Zhao, Z., Mao, Y., Liu, Y., Song, L., Ouyang, Y., Chen, X. & Ding, W. Towards efficient communications in federated learning: A contemporary survey. *J Franklin Inst* **360**, 8669-8703, doi:10.1016/j.jfranklin.2022.12.053 (2023).

[11] Pei, S., Wu, Y. & Qiu, M. Neural network compression and acceleration by federated pruning. *In Algorithms and Architectures for Parallel Processing* (ed. Qiu, M.) **12453**, 173–183 (Springer International Publishing, Cham) (2020).

[12] Khan, F. M. A., Abou-Zeid, H. & Hassan, S. A. Model pruning for efficient over-the-air federated learning in tactical networks. *In 2023 IEEE International Conference on Communications Workshops (ICC Workshops)* **1109**, 1806–1811 (IEEE) (2023).

[13] Dettmers, T. 8-bit approximations for parallelism in deep learning. arXiv preprint arXiv:1511.04561 (2015).

[14] Hu, K., Wu, C. & Zhu, E. HGC: Hybrid gradient compression in distributed deep learning. *In Advances in Artificial Intelligence and Security* (eds. Sun, X., Zhang, X., Xia, Z. & Bertino, E.) **1422**, 15–27 (Springer International Publishing, Cham) (2021).

[15] Alistarh, D., Grubic, D., Li, J. Z., Tomioka, R. & Vojnovic, M. QSGD: Communication-efficient SGD via gradient quantization and encoding. *In Proceedings of the 31st International Conference on Neural Information Processing Systems* **1422**, 1707–1718 (Curran Associates Inc.) (2017).

[16] Lu, Q., Liu, W., Han, J. & Guo, J. Multi-stage gradient compression: Overcoming the communication bottleneck in distributed deep learning. *In Neural Information Processing* (eds. Cheng, L., Leung, A. C. S. & Ozawa, S.) **11301**, 107-119 (Springer International Publishing) (2018).

[17] Hardy, C., Le Merrer, E. & Sericola, B. Distributed deep learning on edge-devices: Feasibility via adaptive compression. *In 2017 IEEE 16th International Symposium on Network Computing and Applications (NCA)* 1-8 (IEEE) (2017).

[18] Jiang, Y., Wang, S., Valls, V., Ko, B. J., Lee, W.-H., Leung, K. K. & Tassiulas, L. Model pruning enables efficient federated learning on edge devices. *IEEE Trans Neural Netw Learn Syst* **34**, 10374-10386, doi:10.1109/TNNLS.2022.3166101 (2023).

[19] Chen, C., Xu, H., Wang, W., Li, B., Li, B., Chen, L. & Zhang, G. Communication-efficient federated learning with adaptive parameter freezing. *In 2021 IEEE 41st International Conference on Distributed Computing Systems* 1-11 (IEEE) (2021).

[20] Jhunjhunwala, D., Gadhikar, A., Joshi, G. & Eldar, Y. C. Adaptive quantization of model updates for communication-efficient federated learning. *In 2021 IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP) 3110-3114 (IEEE) (2021).

[21] Mao, Y., Zhao, Z., Yan, G., Liu, Y., Lan, T., Song, L. & Ding, W. Communication-efficient federated learning with adaptive quantization. *ACM Trans Intell Syst Technol* **13**, 67, doi:10.1145/3510587 (2022).

[22] Haddadpour, F., Karimi, B., Li, P. & Li, X. Fedsketch: Communication-efficient and private federated learning via sketching. *arXiv preprint arXiv*:2008.04975 (2020).

[23] Lin, Y., Han, S., Mao, H., Wang, Y. & Dally, W. J. Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv preprint arXiv*:1712.01887 (2017).

[24] Vogels, T., Karimireddy, S. P. & Jaggi, M. PowerSGD: Practical low-rank gradient compression for distributed optimization. *Adv Neural Inf Process Syst* **32**, (2019).

[25] Malekijoo, A., Fadaeieslam, M. J., Malekijou, H., Homayounfar, M., Alizadeh-Shabdiz, F. & Rawassizadeh, R. Fedzip: A compression framework for communication-efficient federated learning. *arXiv preprint arXiv*:2102.01593 (2021).

[26] Dai, W., Fan, J., Miao, Y. & Hwang, K. Deep learning model compression with rank reduction in tensor decomposition. *IEEE Trans Neural Netw Learn Syst* **36**, 1315–1328, doi:10.1109/TNNLS.2023.3330542 (2025).

[27] Paragliola, G. & Coronato, A. Definition of a novel federated learning approach to reduce communication costs. *Expert Syst Appl* **189**, 116109, doi:10.1016/j.eswa.2021.116109 (2022).

[28] Bellavista, P., Foschini, L. & Mora, A. Communication-efficient heterogeneous federated dropout in cross-device settings. *In 2021 IEEE Global Communications Conference* (GLOBECOM) 1–6 (IEEE) (2021).

[29] Hoshino, Y., Kawakami, H. & Matsutani, H. Communication size reduction of federated learning based on neural ODE model. *In 2022 Tenth International Symposium on Computing and Networking Workshops* (CANDARW) 55–61 (IEEE) (2022).

[30] Wu, L., Jin, Y. & Hao, K. Optimized compressed sensing for communication efficient federated learning. *Knowl Based Syst* **278**, 110805, doi:10.1016/j.knosys.2023.110805 (2023).

[31] ang, Y., Zhang, Z. & Yang, Q. Communication-efficient federated learning with binary neural networks. *IEEE J Sel Areas Commun* **39**, 3836–3850, doi:10.1109/JSAC.2021.3118415 (2021).

[32] Singh, N., Data, D., George, J. & Diggavi, S. SPARQ-SGD: Event-triggered and compressed communication in decentralized optimization. *IEEE Trans Autom Control* **68**, 721–736, doi:10.1109/TAC.2022.3145576 (2023).

[33] Xu, J., Du, W., Jin, Y., He, W. & Cheng, R. Ternary compression for communication-efficient federated learning. *IEEE Trans Neural Netw Learn Syst* **33**, 1162–1176, doi:10.1109/TNNLS.2020.3041185 (2022).

[34] Chai, Z., Chen, Y., Anwar, A., Zhao, L., Cheng, Y. & Rangwala, H. FedAT: A high-performance and communication-efficient federated learning system with asynchronous tiers. In SC '21: *International Conference for High Performance Computing, Networking, Storage and Analysis* 1–17 (ACM) (2021).

[35] Chen, Y., Sun, X. & Jin, Y. Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation. *IEEE Trans Neural Netw Learn* Syst **31**, 4229–4238, doi:10.1109/TNNLS.2019.2953131 (2020).

[36] Itahara, S., Nishio, T., Koda, Y., Morikura, M. & Yamamoto, K. Distillation-based semi-supervised federated learning for communication-efficient collaborative training with non-IID private data. *IEEE Trans Mobile Comput* **22**, 191–205, doi:10.1109/TMC.2021.3070013 (2023).

[37] Jeong, E., Oh, S., Kim, H., Park, J., Bennis, M. & Kim, S.-L. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-IID private data. *arXiv preprint arXiv*:1811.11479 (2018).

[38] attler, F., Marban, A., Rischke, R. & Samek, W. CFD: Communication-efficient federated distillation via soft-label quantization and delta coding. *IEEE Trans Netw Sci Eng* **9**, 2025–2038, doi:10.1109/TNSE.2021.3081748 (2022).

[39] Cheng, S., Wu, J., Xiao, Y. & Liu, Y. Fedgems: Federated learning of larger server models via selective knowledge fusion. *arXiv preprint arXiv*:2110.11027 (2021).

[40] He, C., Annavaram, M. & Avestimehr, S. Group knowledge transfer: Federated learning of large CNNs at the edge. *In Advances in Neural Information Processing Systems* (Curran Associates, Inc.) (2020).

[41] Wu, C., Wu, F., Lyu, L., Huang, Y. & Xie, X. Communication-efficient federated learning via knowledge

distillation. *Nat Commun* **13**, 2032, doi:10.1038/s41467-022-29763-x (2022).
[42] Khan, A., ten Thij, M. & Wilbik, A. Communication-efficient vertical federated learning. *Algorithms* **15**, 273, doi:10.3390/a15080273 (2022).
[43] Sun, J., Xu, Z., Yang, D., Nath, V., Li, W., Zhao, C., Xu, D., Chen, Y. & Roth, H. R. Communication-efficient vertical federated learning with limited overlapping samples. *In *2023 IEEE/CVF International Conference on Computer Vision* (ICCV)* 5180–5189 (IEEE) (2023).
[44] Castiglia, T., Zhou, Y., Wang, S., Kadhe, S., Baracaldo, N. & Patterson, S. LESS-VFL: Communication-efficient feature selection for vertical federated learning. *In Proceedings of the 40th International Conference on Machine Learning (JMLR.org*) (2023).
[45] Yao, X., Huang, T., Wu, C., Zhang, R. & Sun, L. Towards faster and better federated learning: A feature fusion approach. *In 2019 IEEE International Conference on Image Processing (ICIP)* 175–179 (IEEE) (2019).
[46] Yao, X., Huang, C. & Sun, L. Two-stream federated learning: Reduce the communication costs. *In 2018 IEEE Visual Communications and Image Processing (VCIP)* 1–4 (IEEE) (2018).
[47] Waghmare, S., Qi, H., Chen, H., Sirotenko, M. & Meron, T. Efficient image representation learning with federated sampled softmax. *arXiv preprint arXiv*:2203.04888 (2022).
[48] Lang, N. & Shlezinger, N. Joint privacy enhancement and quantization in federated learning. *In 2022 IEEE International Symposium on Information Theory (ISIT)* 2040–2045 (IEEE) (2022).
[49] Nariman, G. S. & Hamarashid, H. K. Communication overhead reduction in federated learning: a review. *Int J Data Sci Anal* **19**, 185–216, doi:10.1007/s41060-024-00691-x (2025).
[50] Deng, L. The MNIST database of handwritten digit images for machine learning research [Best of the Web]. *IEEE Signal Process Mag* **29**, 141–142, doi:10.1109/MSP.2012.2211477 (2012).
[51] McMahan, H. B., Moore, E., Ramage, D., Hampson, S. & y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. *In Artificial Intelligence and Statistics (eds. Gretton, A. & Robert, C. C.)* 1273–1282 (PMLR) (2016).
[52] Kim, M., Yu, S., Kim, S. & Moon, S. M. DepthFL: Depthwise federated learning for heterogeneous clients. *In 11th International Conference on Learning Representations (ICLR)* (2023).
[53] Kolasa, D., Pilch, K. & Mazurczyk, W. Federated learning secure model: A framework for malicious clients detection. *SoftwareX* **27**, 101765, doi:10.1016/j.softx.2024.101765 (2024).
[54] Nilsson, A., Smith, S., Ulm, G., Gustavsson, E. & Jirstrand, M. A performance evaluation of federated learning algorithms. *In Proceedings of the Second Workshop on Distributed Infrastructures for Deep Learning* 1–8 (ACM) (2018).
[55] Briggs, C., Fan, Z. & Andras, P. Federated learning with hierarchical clustering of local updates to improve training on non-IID data. *In 2020 International Joint Conference on Neural Networks (IJCNN)* 1–9 (IEEE) (2020).
[56] Qu, Z., Li, X., Han, X., Duan, R., Shen, C. & Chen, L. How to prevent the poor performance clients for personalized federated learning? *In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)** 12167–12176 (IEEE) (2023).