Proceedings of the 4th International Conference on Mechatronics and Smart Systems
DOI: 10.54254/2755-2721/2026.33435

Communication-Efficient Federated Learning under Non-IID Data: A Comparative Study of Gradient Compression Techniques

Ziyi Wang

School of Information Science and Engineering, East China University of Science and Technology, Shanghai, China
22013164@mail.ecust.edu.cn

Abstract. Federated Learning (FL) has become an important way to train models in a distributed way while protecting data privacy. But the huge communication costs are still the main problem when using it in edge networks with limited bandwidth. This paper does a detailed comparison to see how Top-k sparsification and bit-quantization, two common gradient compression methods, affect model performance and communication efficiency. Using the CIFARNet model on the CIFAR-10 dataset, the author did a lot of methods to make the experiments more like real edge environments. This paper used a Non-IID data splitting method based on the Dirichlet distribution to simulate the uneven distribution of data in reality. The results presents that Top-k sparsification (20%) saves the most bandwidth, cutting communication costs by 60% in total, though it only causes a acceptable drop in accuracy of about 19% compared to the uncompressed baseline. Also, this Top-k rate maintains fast convergence and training stability. As for quantization, 16-bit quantization offers a 3% down in accuracy while cutting communication volume in half. Besides, 8-bit quantization also performs impressively: although it causes a minor unstable oscillations under the broad convergence trend, it brings a 75% cutoff on the total communication with a 3% drop on accuracy in average. In short, while sparsification is good for extreme bandwidth limits because it compresses aggressively, quantization is more stable when dealing with differences in client data in highly heterogeneous settings.

Keywords: federated learning, communication efficiency, top-k sparsification, gradient quantization

1. Introduction

Since more and more people have raised awarenesses on data privacy in this age when AI steps into everyone's daily life, many large model service providers have turned their attention to Federated Learning (FL). Unlike traditional centralized training, FL allows multiple separate clients to collaboratively train a global model without sharing raw local data, meeting strict privacy rules. The well-known Federated Averaging (FedAvg) algorithm laid the foundation: clients regularly update local models, and these updates are combined into a global model. However, due to the heavy

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2026.33435

transmission of model data (e.g. gradients or parameters) over bandwidth-limited wireless networks [1], the cost of communication resources has become one of the biggest problems in FL.

Nowadays, LLM structures have become more complex, which can reach hundreds of megabytes, making frequent synchronization costly in terms of latency and energy. To solve this, researchers have explored ways to improve communication efficiency. In 2016, Konečný et al. categorized these methods into structured and sketched updates, paving the way for better gradient compression techniques [2]. They carried out that gradient quantization and sparsification are the most important. Gradient quantization reduces the bits used to represent each update element; for instance, QSGD uses random quantization to balance communication cost and convergence speed, with a provable trade-off [3]. Gradient sparsification (e.g. Top-k) is a kind of method which only transmits the most important gradient elements and ignores the rest. In 2017, Lin et al. used techniques like momentum correction in their experiment [4]. They showed deep gradient compression can reduce bandwidth by up to 1000 times with little accuracy loss. However, sparsification often causes biased updates that may derail training without proper handling, requiring error feedback mechanisms that store untransmitted gradients for subsequent rounds [5]. Recent work has also combined Top-k sparsification with secure aggregation to enhance privacy while maintaining efficiency (Lu et al., 2022) [6].

Even worse, it is more challenging to do compression in real FL environments due to inherent data heterogeneity. According to Li et al., different data distributions across clients can cause local models to stray far from the global optimum, and compression tends to amplify this issue [7]. While both Top-k sparsification and bit-quantization save significant bandwidth [8], their impact on accuracy and convergence stability in highly heterogeneous settings remains under intense investigation. To provide a solution for this problem, this paper aims to present a practical guidance for selecting optimal strategies based on network constraints. The author uses the CIFAR-10 dataset and Dirichlet-based data partitioning to evaluate how Top-k sparsification and bit-quantization affect the balance between accuracy and communication cost in a Non-IID framework.

2. Method

2.1. Dataset preparation

This paper uses the CIFAR-10 dataset [9], a commonly used standard for image classification tasks. Being split evenly into 10 different categories: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck, the dataset has 60,000 RGB color images, each 32×32 pixels in size. These images are divided into 50,000 for training and 10,000 for testing, which means each category has 5,000 training images and 1,000 test images. Researchers have used many different convolutional neural network structures to do image classification on this dataset (Lv, 2020) [10]. Figure 1 shows sample images from the CIFAR-10 dataset, which shows how diverse and challenging the classification task is.

DOI: 10.54254/2755-2721/2026.33435

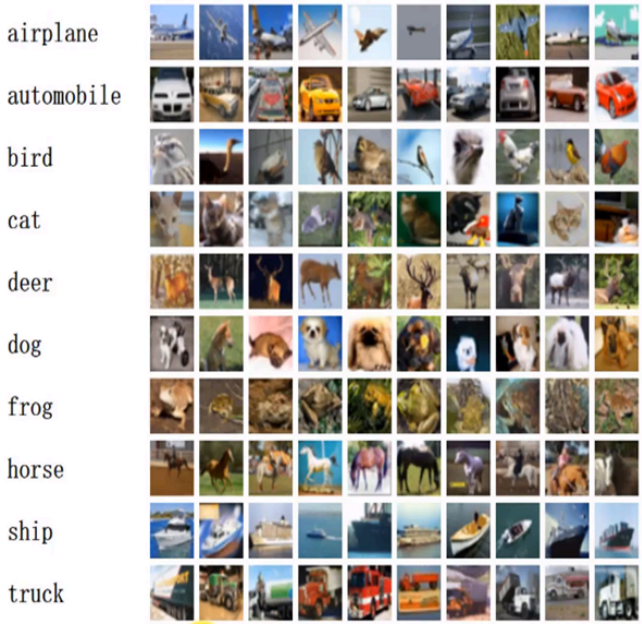


Figure 1. CIFAR-10 sample images [9]

Before training, data augmentation techniques (e.g. randomly cropping images with a 4-pixel padding, flipping images horizontally at random) are used in training set to processes the images to help the model work better on new, unseen data. All images are normalized using channel-wise mean and standard deviation values: $\mu = (0.4914,0.4822,0.4465)$ and $\sigma = (0.247,0.243,0.261)$ respectively. The test set is only normalized, with no augmentation.

To simulate real-world federated learning situations where data is not evenly distributed, this paper uses a Non-Independent and Identically Distributed (Non-IID) data splitting method based on the Dirichlet distribution [11]. The author split the dataset among 5 clients using a concentration parameter $\alpha = 0.5$, which makes the data distribution across clients very different. This splitting method assigns each client an imbalanced class distribution, reflecting real-world conditions in which data varies considerably across devices. Compared to traditional IID settings, this Non-IID framework is a stricter test for how well compression algorithms work.

2.2. Federated learning with gradient compression

The basic Federated Averaging (FedAvg) algorithm sets the foundation for this process: in each communication round, chosen clients train the model locally on their own data, calculate updates for the model, and send these updates to a central server, Then the server combines all the received updates by weighted averaging, sends the updated global model back to the clients for the next round of training [1].

This flow works well on models having small gradients or parameters. However, while facing with models with high-dimensional parameters, a big problem with this approach is the large amount of communication needed to send. For modern deep neural networks, a single model update can be more than several megabytes. Since frequent synchronization over many rounds is needed, the total communication cost becomes too high, especially in edge computing environments where bandwidth is limited.

To have a better understanding on this problem, this study compares two popular gradient compression techniques: Top-k sparsification and bit-quantization. These methods are applied on the

DOI: 10.54254/2755-2721/2026.33435

client side before sending updates, which can reduce the size of the model updates.

Top-k Sparsification is a approach which only sends the $k\%$ of gradient elements with the largest absolute values, which effectively makes the data sparse. Mathematically, for a gradient vector $g \in R^d$, the compressed version $\widehat{\boldsymbol{g}}$ is computed as:

$$\widehat{\boldsymbol{g}}_i \begin{cases} g_i, \ if \ |g_i| \geq \tau \\ 0, \ otherwise \end{cases} \tag{1}$$

Here, τ_k represents the $(100-p)$ -th percentile of $\{|g_i|\}_{i=1}^{d}$ (i.e., exactly $p\%$ of the entries have $|g_i| \geq \tau$). This paper composes a compression algorithm which utilizes argpartition for efficient identification of top-k elements, transmitting only their indices and corresponding values. When decompressing, the server rebuilds the full gradient tensor by putting the received values in their correct positions and filling the rest with zeros. This paper evaluates Top-k with retention ratios of 5%, 10%, and 20%.

Bit-Quantization means reducing the precision of gradient elements by using fewer bits to represent each element. This paper tests two types: 16-bit and 8-bit quantization. The 16-bit method converts float32 gradients to float16 format, cutting the size in half with almost no loss in precision. The 8-bit is a little bit more complex than 16-bit: it uses linear scaling, mapping gradient values to the int8 range [-127, 127] with a scaling factor $s = \frac{max(|g|)}{127}$. Both methods need to send scaling information along with the quantized values to make sure the server can rebuild the data accurately.

Figure 2 and Figure 3 show the whole federated learning process with gradient compression. The process starts by splitting the data among clients using the Dirichlet distribution. Then, in each training round, local models calculate compressed gradients and send them to the server. The server combines these updates and sends the improved global model back to the clients. This cycle continues until the model converges or finishes the planned number of rounds.

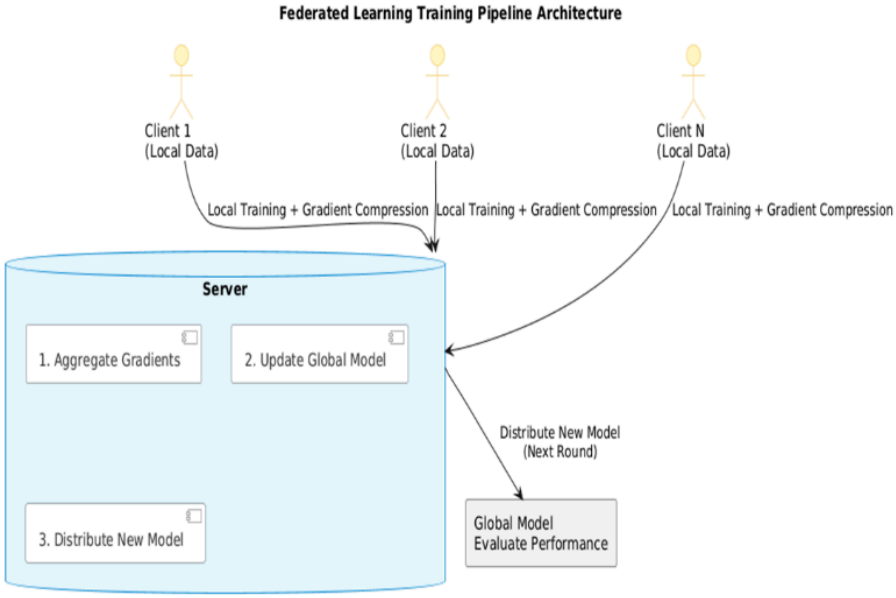


Figure 2. Training pipeline architecture (picture credit: original)

DOI: 10.54254/2755-2721/2026.33435

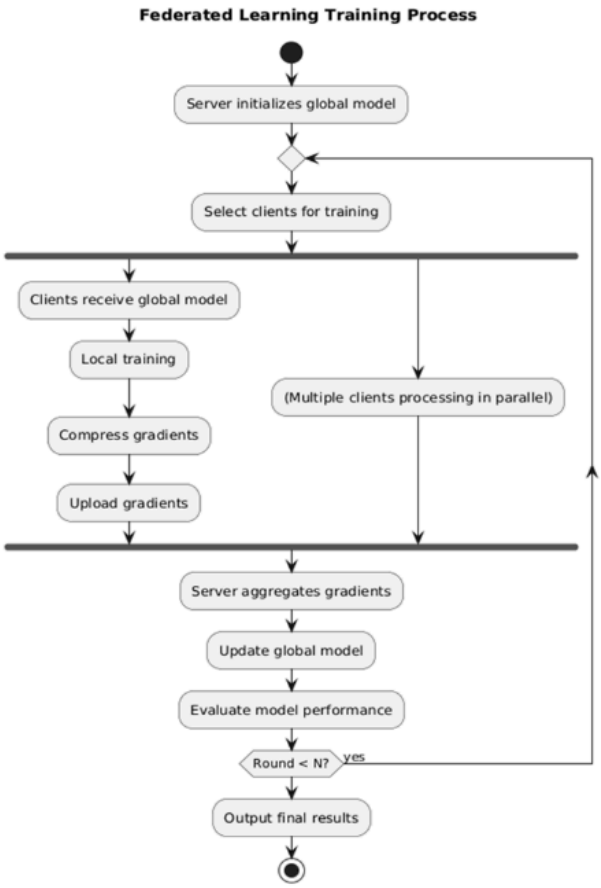


Figure 3. Training workflow (picture credit: original)

2.3. Experimental hyperparameter configuration

The experiments use a custom convolutional neural network (CIFARNet) that's optimized for the CIFAR-10 classification task. As Figure 4 shows, the network has three convolutional blocks with Batch Normalization, followed by two fully connected layers with Dropout regularization (set to 0.5). The model has about 620,000 parameters that can be trained.

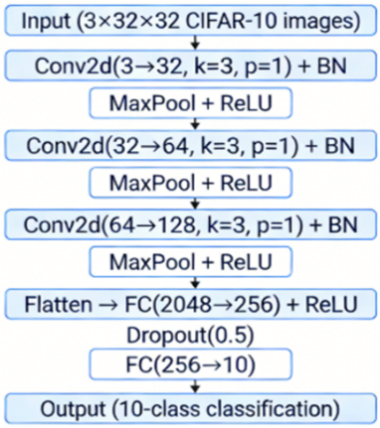


Figure 4. CIFARNet architecture used in this article: the structure is inspired by model caffe [12]

DOI: 10.54254/2755-2721/2026.33435

For local training, each client uses the Stochastic Gradient Descent (SGD) optimizer with momentum. The experimental parameters are set as follows: learning rate $\eta = 0.01$, momentum coefficient $\beta = 0.9$, weight decay $\lambda = 1 \times 10^{-4}$, and a batch size of 32. Each client trains the model locally for 2 epochs per communication round before sending updates to the server.

The federated learning setup has 5 clients participating in each round, over a total of 10 communication rounds. All clients take part in every round (fraction_fit=1.0) to make sure it is possible to compare the compression methods fairly. This paper evaluates the model using three metrics: test accuracy, test loss, and total communication cost.

To make sure the experiments can be repeated, all tests use a fixed random seed (seed=42). The Non-IID data distribution is controlled by the Dirichlet concentration parameter $\alpha = 0.5$, which makes the data across clients very different. This setup is challenging for compression algorithms because local models trained on uneven data may produce gradients that drift apart and aggressive compression could make this problem even worse.

3. Results and discussion

3.1. Experimental results

This study did detailed experiments to compare six different setups: no compression (used as a baseline), Top-k sparsification with 5%, 10%, and 20% retention rates, and bit-quantization with 16-bit and 8-bit precision. Table 1 shows the final test accuracy, the best accuracy achieved during training, total communication cost, and average communication cost per round for each method.

Table 1. Performance comparison of compression methods in Non-IID federated learning

Method	Final Accuracy	Best Accuracy	Total Comm. (MB)	Avg. Comm./Round (MB)
No Compression	0.700	0.700	111.39	11.139
Top-k (5%)	0.100	0.100	11.15	1.115
Top-k (10%)	0.457	0.457	22.29	2.229
Top-k (20%)	0.569	0.569	44.56	4.456
Quantize (16bit)	0.678	0.678	55.71	5.571
Quantize (8bit)	0.651	0.702	27.86	2.786

As the result of the experiment shows, the uncompressed baseline has the smoothest convergence trend, reaching a competitive accuracy by the 10th round. In contrast, all Top-k methods significantly slow down convergence speed. Among them, Top-k 20% is closest to the baseline, while Top-k 10% falls behind moderately. However, Top-k sparsification at the 5% level exhibits nearly no convergence in this experimental setting despite achieving substantial communication reduction. As for quantization methods, 16-bit quantization maintains performance almost as good as the baseline while 8-bit quantization shows obvious accuracy fluctuations throughout training. Figure 5 intuitively illustrates how accuracy and loss evolve over all 10 training rounds and reveals clear differences in the convergence patterns of each compression method.

DOI: 10.54254/2755-2721/2026.33435

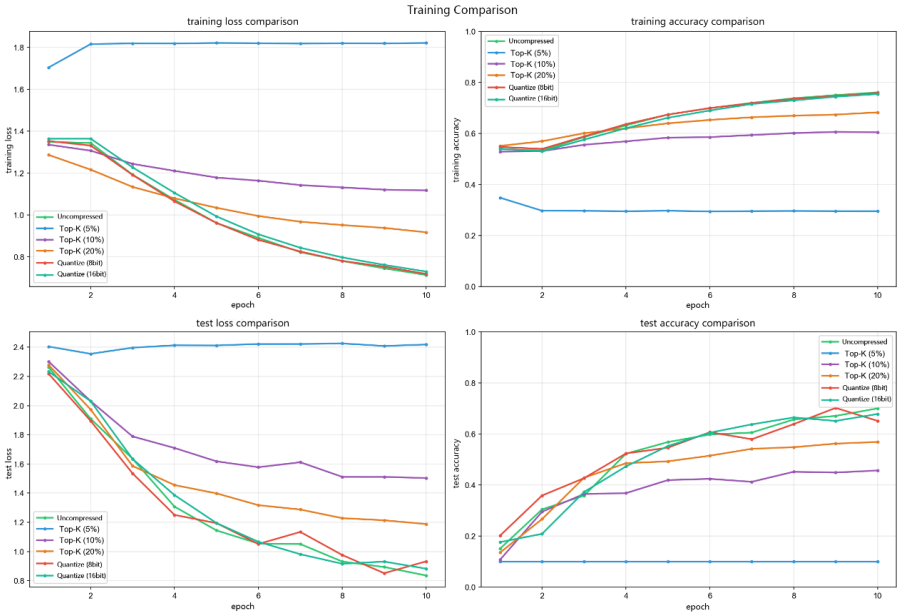


Figure 5. Accuracy and loss comparison (picture credit: original)

Figure 6 compares the accuracy of tests and the cost of communication for models that use different algorithms. The baseline collects about 111.39 MB over 10 rounds, but the compression methods save a lot of space. In the worst cases, Top-k methods can cut communication by the most, with only 11.15 MB needed to keep 5% of the data (90% savings compared to the baseline). Even with a 20% retention rate, the communication cost can still be cut by 60%. Quantization methods save less money than the baseline. For example, 16-bit quantization only needs 50% of the baseline, and 8-bit quantization only needs 25% of the baseline. However, their final convergence speed and model accuracy do not drop significantly compared to the baseline.

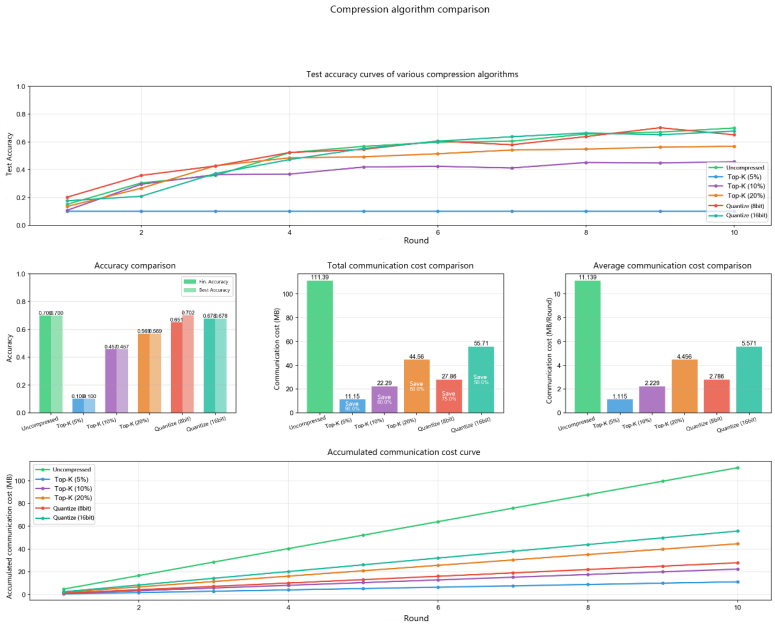


Figure 6. Communication cost and test accuracy comparison (picture credit: original)

DOI: 10.54254/2755-2721/2026.33435

3.2. Discussion

The experiment findings yield significant insights into the equilibrium between communication efficiency and model accuracy in Non-IID federated learning. The results indicate that gradient sparsification using Top-k 5% yields the greatest bandwidth savings; however, this is accompanied by considerable accuracy deterioration in a highly heterogeneous data environment. The big data skew makes local models produce different gradients, and aggressive sparsification can get rid of important information that is needed for global convergence. This is in line with what Li et al. found, which showed that compression can make the bad effects of data heterogeneity worse [7].

The results show that Top-k 20% is relatively the best way to balance compression and accuracy in this experiment. It gets rid of 80% of the gradient elements to save 60% on communication costs while still keeping accuracy competitive (within 81.3% of the baseline). This means that moderate sparsification can keep enough gradient information for convergence in Non-IID settings. Its better performance than Top-k 10% shows that there is a non-linear relationship between sparsification ratio and accuracy, with diminishing returns after certain points.

However, in this situation, bit-quantization methods work better than sparsification. 16-bit quantization keeps accuracy close to the baseline while cutting communication in half, making it perfect for real-world use. Even though 8-bit quantization saves 75% of bandwidth, it has big problems with accuracy because reducing precision too much makes it harder to converge when the data is different.

In short, when comparing the two compression algorithms, quantization works better in Non-IID settings because it has better accuracy-communication trade-offs, while extreme Top-k saves more bandwidth overall. For many applications, extreme sparsification (5%) may have unacceptable accuracy loss whereas 16-bit quantization provides reliable 50% savings with minimal loss and 8-bit offers 75% savings with acceptable degradation for some use cases. Also, even small compression ratios can save a lot of money over time in long-running deployments. For instance, Top-k 20% saves about 66.83 MB over 10 rounds and 668.3 MB over 100 rounds.

These results can help people choose compression strategies in Non-IID federated learning over time. Quantization is very useful when focusing on applications where accuracy is more important than bandwidth. Top-k can help save money in situations where it is required to cut bandwidth by a lot. Also, it is required to be careful when using aggressive sparsification and extreme quantization because they can cause convergence instability.

4. Conclusion

This paper examines the balance between communication efficiency and model performance in federated learning with Non-IID data distributions. The results of a thorough comparison of Top-k sparsification and bit-quantization on the CIFAR-10 dataset show that the right kind of gradient compression can lower communication costs significantly while still keeping accuracy at an acceptable level. Specifically, aggressive Top-k sparsification saves a lot of bandwidth, but it hurts accuracy because the gradient updates aren't even, especially in environments with a lot of different types of data. Quantization methods, especially 16-bit quantization, on the other hand, provide more stable training and are better at handling client data that is not all the same. They strike a good balance between efficiency and performance.

The results show that when setting up federated learning systems, it is important to choose compression strategies carefully based on the needs of the application. Moreover, subsequent investigations in this domain may examine adaptive compression techniques, extensive client

DOI: 10.54254/2755-2721/2026.33435

configurations, and sophisticated methodologies such as error feedback to enhance training stability and scalability in practical federated environments.

References

[1] McMahan, B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In Artificial intelligence and statistics (pp. 1273-1282). Pmlr.
[2] Konečný, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., & Bacon, D. (2016). Federated learning: Strategies for improving communication efficiency. arXiv preprint arXiv: 1610.05492.
[3] Alistarh, D., Grubic, D., Li, J., Tomioka, R., & Vojnovic, M. (2017). QSGD: Communication-efficient SGD via gradient quantization and encoding. Advances in neural information processing systems, 30.
[4] Lin, Y., Han, S., Mao, H., Wang, Y., & Dally, W. J. (2017). Deep gradient compression: Reducing the communication bandwidth for distributed training. arXiv preprint arXiv: 1712.01887.
[5] Stich, S. U., Cordonnier, J. B., & Jaggi, M. (2018). Sparsified SGD with memory. Advances in neural information processing systems, 31.
[6] Lu, S., Li, R., Liu, W., Guan, C., & Yang, X. (2023). Top-k sparsification with secure aggregation for privacy-preserving federated learning. Computers & Security, 124, 102993.
[7] Li, T., Sahu, A. K., Talwalkar, A., & Smith, V. (2020). Federated learning: Challenges, methods, and future directions. IEEE signal processing magazine, 37(3), 50-60.
[8] Singh, N., Data, D., & Diggavi, S. (2023). Communication Efficient Distributed Learning. In Artificial Intelligence for Edge Computing (pp. 199-222). Cham: Springer International Publishing.
[9] Krizhevsky, A., & Hinton, G. (2009). Learning multiple layers of features from tiny images.
[10] Lv, X. Y. (2020). CIFAR-10 image classification based on convolutional neural network. Frontiers in Signal Processing, 4(4), 100-106.
[11] Hsu, T., Qi, H., & Brown, M. (2019). Measuring the effects of non-iid data in federated learning. In Proceedings of the 1st Workshop on Federated Learning for Data Privacy and Confidentiality, Vancouver, BC, Canada (Vol. 13).
[12] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., ... & Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. In Proceedings of the 22nd ACM international conference on Multimedia (pp. 675-678).