Federated learning compression designed for lightweight communications

Lucas Grativol Ribeiro, Mathieu Leonardon, Guillaume Muller, Fresse, Virginie, Matthieu Arzel

Federated learning compression designed for lightweight communications

Lucas Grativol*, Mathieu Léonardon*, Guillaume Muller‡, Virginie Fresse† and Matthieu Arzel*
*IMT Atlantique, Lab-STICC, UMR CNRS 6285, F-29238 Brest, France
†Hubert Curien Laboratory, Saint-Etienne, France
‡Mines Saint-Etienne, Institut Henri Fayol, Saint-Etienne, France

***Abstract*—Federated Learning (FL) is a promising distributed method for edge-level machine learning, particularly for privacy-sensitive applications such as those in military and medical domains, where client data cannot be shared or transferred to a cloud computing server. In many use-cases, communication cost is a major challenge in FL due to its natural intensive network usage. Client devices, such as smartphones or Internet of Things (IoT) nodes, have limited resources in terms of energy, computation, and memory. To address these hardware constraints, lightweight models and compression techniques such as pruning and quantization are commonly adopted in centralised paradigms. In this paper, we investigate the impact of compression techniques on FL for a typical image classification task. Going further, we demonstrate that a straightforward method can compresses messages up to 50% while having less than 1% of accuracy loss, competing with state-of-the-art techniques.**

***Index Terms*—Compression, Federated Learning, Embedded Systems**

I. Introduction

The development of approaches for training machine learning models while preserving data privacy has long been a goal. In traditional machine learning, embedded systems send their raw data over a network to a powerful server, which then trains the model and sends it back. However, this process raises confidentiality issues, such as data interception during communication and unauthorised access to user data by the server owner or a third party. In standard Federated Learning (FL), the server sends a model to a group of clients, who train it on their local data and then send their updated parameters back to the server for aggregation. By reversing the training process in this way, FL attempts to better guarantee the confidentiality of user data, since data never leaves a client device. An overview of the process can be seen in Fig. 1.

These embedded devices such as IoT devices, smartphones and drones are well suited to FL applications due to their proximity to real-world data and applications [1]. However, many of these devices have limited computational resources and co-design techniques [2] are continually being explored to match algorithms to hardware constraints. Among the emerging research topics for FL at the edge/device level, the field of neural network compression is a promising way to tackle the constraints of devices exploiting FL [3].

This work is supported by the *Futur et Ruptures* program funded by IMT and Institut Carnot TSN, and by the GdR ISIS.

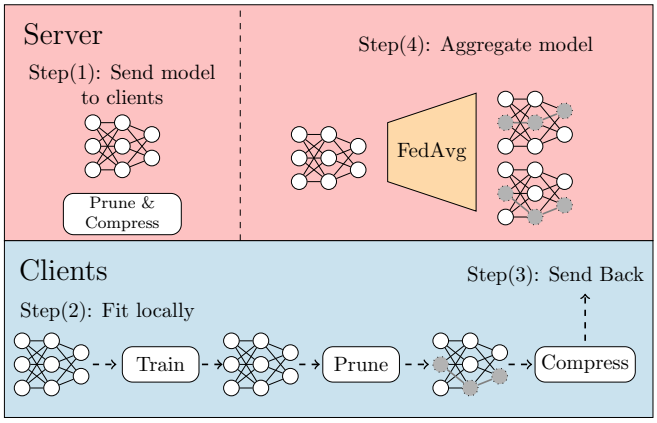

Fig. 1: The pipeline of our study. We propose a simple way to insert the pruning technique as extra step before communicating training results.

In addition to message compression, the FL domain encompasses important ongoing research efforts. These include addressing challenges related to client heterogeneity in terms of both data and hardware [4], ensuring secure aggregation against attacks [5], and increasing client's privacy [3]. While our work primarily focuses on reducing message sizes for energy and bandwidth reductions, we emphasize the importance of seamless integration with other ongoing research in FL. When compared to previous approaches [6], [7], we propose a simpler and more effective solution that not only reduces message sizes but also ensures the possibility to be combined with other techniques without compromising accuracy. Our code is publicly accessible [1].

II. Background

A. Overview of Federated Learning

Federated Learning (FL) [8] is a distributed framework that enables collaborative training of machine learning (ML) models on multiple devices, called clients, via a central coordinator, usually a server with large compute resources. Clients are commonly embedded devices, such as smartphones. Differing from traditional ML, each client trains its own local model and shares only the local training results, like model parameters or gradients, with the server. Through this mechanism, multiple

[1] https://github.com/lgrativol/fl_exps

clients can jointly contribute to train a global model without sharing their data. In each federated training cycle, commonly referred to as a 'round,' the server distributes the current model to a subset of clients, who perform local training on the model and subsequently send back the updated results. The final step involves aggregating, on the server-side, client's contribution to create a global model, which ideally can represent the knowledge from each client. Each round involves downloading the model and several training iterations at client level.

B. Model Compression

Model compression is a widely adopted solution [2] to reduce the computational and memory requirements of a model. Among existing compression techniques, quantization and pruning have been implemented to reduce the complexity of inference and training of neural networks [9], [10].

Pruning aims to reduce the complexity of a model by removing redundant or unnecessary parts of an architecture. Very wide and deep models tend to yield good results, but the contribution of each of its elements to the performance of the whole network is not homogeneous. So, by observing each architectural element of the network, it is possible to eliminate those that have little impact. There are two possible approaches to pruning in the context of neural networks [11]. The first is to replace the value of certain weights with zeros, which is commonly referred to as unstructured pruning. The second approach consists of pruning entire structures within the network, such as kernels, filters or layers, which do not contribute significantly to the network's performance. This approach is known as structured pruning. On the other hand, unstructured pruning can also offer compression benefits for FL through the use of entropy coding techniques, such as Huffman [12] coding, by exploiting sparse parameters. So far works in the literature [6] have used pruning to reduce communication cost by an order of 4.5 times. This is a significant consideration since typical FL clients often encompass low-power devices and operate in challenging transmission environments, such as long-distance or underwater communications.

Another widely applied technique is quantization, neural network models are generally constructed using 32-bit floating point numbers (FP32), which are more expensive in terms of computation, memory and energy than integers [13]. In centralized machine learning is well-known that full-precision, FP32, it's not a necessary condition to obtain close to state-of-the-art results for inference and training [12].

ZeroFL [6] is a recent work that seeks to reduce simultaneously communication and training costs with a double optimization scheme to FL. First, a sparse training method named SWAT [14], and second, a layer-wise pruning based on weight importance. However as shown in [7] communications cost can be much higher than the training cost. What should be done in the case where the focus is solely on communication costs ?

III. Magnitude Pruning for Federated Learning

We address the invoked problem in II by proposing a distributed non-structured pruning method. Unlike previous works, our objective is to demonstrate that the conventional FL framework can be modified to support sparse messages. This method results in a compression of approximately 50% of the original size while preserving accuracy with less than a 1% loss. Our implementation is streamlined and easily extendable, making it compatible with more advanced FL algorithms.

Starting from the standard FL pipeline, we introduced pruning as a way to sparsify messages, server to client and vice-versa. Inspired by [12], both server and clients perform a non-structured magnitude pruning just before transmitting a message. This pruning method is based on pruning the absolute value of the global weights following a predetermined pruning rate. Accordingly, the $\theta\%$ smaller weights are substituted with zeros, thereby pruned. Consequently, both the server and clients attain an equivalent level of sparsity in the message throughout each round.

At first, we conducted experiments to study the behaviour of our method while taking into account the impact on message compression. We applied different levels of pruning to detect trade-offs between compression and FL training mechanisms according to the experimental setup illustrated in Fig. 1. Building upon the results of our experiment, we extended our study to include a comparison with a recently published paper to showcase the viability of our approach.

IV. Experiments

A. Exploring Magnitude Pruning

To explore the impacts of our technique in FL, we simulated an image classification task, on the CIFAR-10 dataset, using a ResNet-12 with 780K parameters and 2.97 MB. We used the Flower [15] framework to simulate 10 FL clients. At each round, 40% of the clients are selected for the training process, and 100 rounds were performed to study the evolution of the server model accuracy. Each client used SGD with momentum as the optimizer. For simplicity, we use the same hyperparameters as [4], also replacing the batch-norm layer by a group-norm layer. The server uses FedAvg as the aggregation strategy. Training examples are distributed across clients with a Latent Dirichlet Allocation (LDA) [16] on the original training set. The LDA partition is controlled by a distribution parameter, α. A smaller value of α results in a more non-IID task, making it more challenging. We examined the behavior of our technique in a relatively IID (Independent and Identically Distributed) scenario with $\alpha = 100$. In this setting, clients possess examples of all the classes.

As noted in previous works [3], [8] the number of local iterations performed by clients during training can have an important impact on model aggregation. For so, we decided to investigate this behaviour in the presence of model compression. The results on Fig. 2 show that spending more time on each client contributes to a more robust model, allowing sparser data communications while retaining approximately

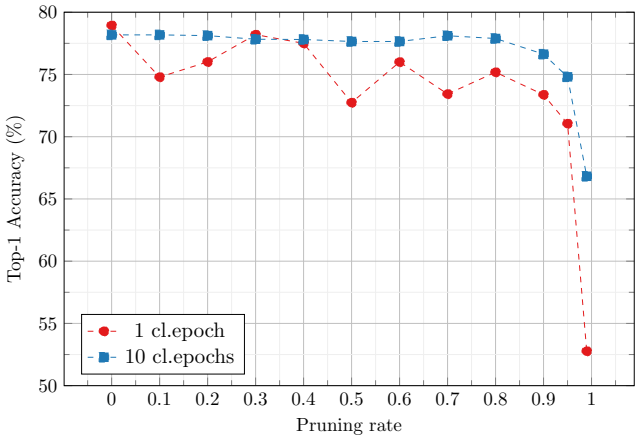

Fig. 2: Pruning effect on the accuracy in function of the pruning rate, where the rate represents the % of total parameters pruned, for 1 and 10 clients epochs.

TABLE I: Comparation to ZeroFL

Method	Compression	Accuracy	Message Size (MB)
ZeroFL	**Full model**	80.62 ± 0.72	**44.7**
	90 % SP + 0.2 Mask Ratio	81.04 ± 0.28	27.3
	90 % SP + 0.0 Mask Ratio	73.87 ± 0.50	10.1
Global magnitude (Ours)	**Full model**	84.43 ± 0.36	**44.7**
	10 % pruning rate	85.96 ± 0.37	38.1
	20 % pruning rate	85.57 ± 0.19	34.8
	30 % pruning rate	85.03 ± 0.32	31.1
	40 % pruning rate	85.20 ± 0.20	27.1
	50 % pruning rate	83.85 ± 0.65	23.0
	60 % pruning rate	83.19 ± 0.44	18.9
	70 % pruning rate	82.25 ± 0.63	14.5
	80 % pruning rate	80.70 ± 0.24	9.8
	90 % pruning rate	76.77 ± 0.47	4.9
	95 % pruning rate	69.14 ± 0.85	2.5
	99 % pruning rate	0.10 ± 0.0	0.5

the same accuracy, even though this approach also results in a higher total number of local iterations.

To further investigate and evaluate the feasibility of our method in a non-IID scenario, we adopted the same test case as ZeroFL. The model is a ResNet-18 with 11M trainable parameters, occupying 44.7 MB. The FL scenario simulates 100 clients with 10% participation rate, for only 1 local epoch and with $\alpha = 0.1$, where clients don't have access to all classes and the number of examples is randomly distributed. Table I presents the model evaluation results. The reported results are the means of three separate runs, with different seeds applied to generate distinct distributions of clients' data. Unless otherwise stated, the size of the models is reported after being compressed using a ZIP algorithm.

In Table I, we present a comparison with ZeroFL [6]. Initially, without pruning, our baseline has a higher accuracy than ZeroFL and as far as we have understood, there are two main distinctions. Firstly, we do not employ SWAT for local training. Secondly, we use a batch size of 8, whereas ZeroFL does not indicate the specific batch size used. As previously observed in FL [8], the batch size is a crucial hyperparameter that influences the aggregation accuracy. Even though SWAT plays a significant role in reducing the communication cost, it also has an impact on the model accuracy, resulting in an overall hindrance. This effect can be noticed as our baseline, which uses pure FedAvg without any compression, already achieves higher accuracy, 4%, when compared to ZeroFL. We observe that for the same level of pruning, our approach exhibits proportionally less degradation. For instance, while ZeroFL experiences an 8% accuracy degradation to prune the model to 10 MB, we only experience a 4.63% degradation. As the results show, client's flexibility to perform pruning on its own better compensates for the sparsity introduced. This compensation enables messages to be more sparse while resulting in a more robust global model.

B. Compressing more with Quantization

From the message savings observed with the pruning experiments, one could ask if it is possible to have even smaller messages. As exposed in section II-B another well-known technique for compression is quantization. In Fig 3, we show the impact of Quantization-Aware Training (QAT) [10], [17] in the IID scenario described before. During QAT, weights are still represented as floating-point numbers but are limited to power-of-two values. At each gradient update, the values are re-evaluated and scaled. The motivation behind using QAT is to incorporate quantization noise into the training procedure, allowing the network to learn from it. We chose to work with 1-bit, 4-bit and 8-bit quantization levels, using Binary Connect [18] for binary networks and the Brevitas [17] framework for 4- and 8-bit with the default quantization scheme. The weights are quantized to 4-bit and 8-bit integers and the QAT scaling is calculated per layer.

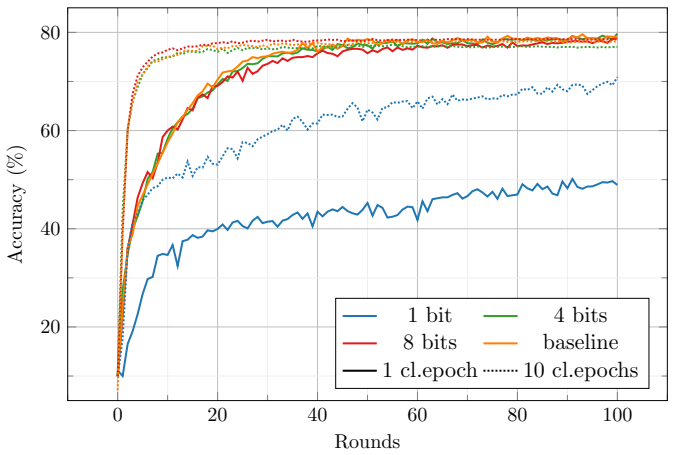

Fig. 3: Accuracy evolution comparison between baseline (32-bit FP), 1-bit, 4-bit and 8-bit, for 1 and 10 clients epochs.

Looking at Fig. 3, we can see that the convergence time, i.e. the number of rounds needed to reach maximum accuracy, is not the same from one experiment to another, as it also depends on the level of quantization. In addition to the fact

TABLE II: Summary of message size and accuracy for the CIFAR-10 dataset for the IID case

Compression Technique	**Accuracy (%)**		**Message Size (MB)**
	1 Local Epoch	10 Local Epochs	
Baseline	78.94	78.18	2.97
Pruning			
10 %	74.79	78.18	2.57
20 %	76.01	78.12	2.34
30 %	78.20	77.83	2.10
40 %	77.50	77.81	1.85
50 %	72.74	77.65	1.57
60 %	76.00	77.65	1.29
70 %	73.43	78.11	1.01
80 %	75.18	77.89	0.70
90 %	73.37	76.63	0.37
95 %	71.05	74.81	0.19
99 %	52.77	66.82	0.04
Quantization			
8 bits	78.80	78.58	0.75
4 bits	79.74	77.04	0.38
1 bit	48.93	70.89	0.10

that the 4- and 8-bit format enables us to achieve an accuracy comparable to the reference, it also reveals a compromise between communication and computation. In order to achieve a similar accuracy of around 75%, Fig. 3, it is necessary to perform 40 rounds of communication and 40 total epochs when using 1 local epoch, while in the case of 10 local epochs, 100 total epochs are needed within 10 rounds. Still, in the case of one bit, increasing the number of epochs per round on the client from 1 to 10 considerably increases accuracy, from 48.8 % to 70.9 %, with the total number of epochs increasing from 100 to 1000, with the same communication cost. As seen in the IID pruning experiment in Fig. 2, spending more time on each client contributes to a more robust model to the perturbations introduced by the quantization.

Table II summarises the size of a message exchanged between client and server for IID scenario. For quantization, the message size depends only on the quantized weights, since the server knows the client's quantization. Table II also shows that even simple approaches can be used to compress a network, representing savings of 2 to 4 times in bandwidth without significantly affecting accuracy.

V. Conclusion

Federated learning represents a new approach to training models in a distributed manner, bringing forth fresh optimization challenges due to the presence of embedded systems serving as FL clients. These clients operate with limited hardware, energy, and communication resources. In this article, we demonstrated the promising application of traditional neural network compression methods in the context of FL. Our easy to implement yet effective technique achieved up to a 50% reduction in message size without any significant impact on accuracy, thereby resulting in direct savings in energy and bandwidth costs. Moreover, our method allows each client to customize their pruning process, enabling greater flexibility to adapt to their unique datasets. By integrating quantization into the training process, we introduced an additional compression technique to the framework. It is conceivable that combining quantization and pruning could further enhance message compression, although our results already demonstrate the significance of both techniques individually. Based on these findings, we posit that incorporating a compression-aware training method, while ensuring seamless integration, is a crucial step in advancing the field of FL.

References

[1] P. P. Ray, "A review on tinyml: State-of-the-art and prospects," *Journal of King Saud University-Computer and Information Sciences*, 2021.

[2] Y. Cheng and et al, "A survey of model compression and acceleration for deep neural networks," *arXiv:1710.09282*, 2017.

[3] P. e. a. Kairouz, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[4] S. e. a. Reddi, "Adaptive federated optimization," *arXiv:2003.00295*, 2020.

[5] H. U. Manzoor, M. S. Khan, A. R. Khan, F. Ayaz, D. Flynn, M. A. Imran, and A. Zoha, "Fedclamp: An algorithm for identification of anomalous client in federated learning," in *2022 29th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. IEEE, 2022, pp. 1–4.

[6] X. e. a. Qiu, "Zerofl: Efficient on-device training for federated learning with local sparsity," *arXiv preprint arXiv:2208.02507*, 2022.

[7] P. Li, G. Cheng, X. Huang, J. Kang, R. Yu, Y. Wu, and M. Pan, "Anycostfl: Efficient on-demand federated learning over heterogeneous edge devices," *arXiv preprint arXiv:2301.03062*, 2023.

[8] B. e. a. McMahan, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[9] T. e. a. Hoefler, "Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks." *J. Mach. Learn. Res.*, vol. 22, no. 241, pp. 1–124, 2021.

[10] J. e. a. Lin, "On-device training under 256kb memory," *arXiv:2206.15472*, 2022.

[11] H. e. a. Tessier, "Rethinking weight decay for efficient neural network pruning," *Journal of Imaging*, vol. 8, no. 3, p. 64, 2022.

[12] S. e. a. Han, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[13] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, 2014, pp. 10–14.

[14] M. A. Raihan and T. Aamodt, "Sparse weight activation training," *Advances in Neural Information Processing Systems*, vol. 33, pp. 15 625–15 638, 2020.

[15] D. J. e. a. Beutel, "Flower: A friendly federated learning research framework," *arXiv:2007.14390*, 2020.

[16] T.-M. H. e. a. Hsu, "Measuring the effects of non-identical data distribution for federated visual classification," *arXiv:1909.06335*, 2019.

[17] A. Pappalardo, "Xilinx/brevitas," 2021. [Online]. Available: https://doi.org/10.5281/zenodo.3333552

[18] M. e. a. Courbariaux, "Binaryconnect: Training deep neural networks with binary weights during propagations," *Advances in neural information processing systems*, vol. 28, 2015.