Compression de réseaux de neurones pour l'apprentissage fédéré

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

Compression de réseaux de neurones pour l'apprentissage fédéré

Lucas GRATIVOL[1] Mathieu LÉONARDON[1] Guillaume MULLER[2] Virginie FRESSE[3] Matthieu ARZEL[1]

[1]IMT Atlantique, Lab-STICC, UMR CNRS 6285, F-29238 Brest, France

[2]Laboratoire Hubert Curien, Saint-Etienne, France

[3]Mines Saint-Etienne, Institut Henri Fayol, Saint-Etienne, France

Résumé – L'Apprentissage Fédéré (AF) est une méthode distribuée pour l'apprentissage automatique au niveau *edge*, utile pour les applications sensibles à la confidentialité. Les dispositifs clients ayant des ressources limitées, tels que les portables ou l'Internet des objets (IoT), doivent limite au maximum la transmission des données pour mieux utiliser leurs ressources. Nous avons étudié l'impact des techniques de compression sur l'AF pour la classification d'images avec le jeu de données CIFAR10 et une architecture ResNet-12, démontrant que ces techniques réduisent les coûts de communication tout en maintenant une précision raisonnable.

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. The communication cost is a major challenge in FL due to its 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 quantization and pruning are commonly adopted. In this study, we investigate the impact of compression techniques on FL for image classification using CIFAR10 and a ResNet-12 architecture. Our experimental results demonstrate the effectiveness of compression techniques in reducing communication costs while maintaining reasonable accuracy.

1 Introduction

Le développement d'approches permettant d'entraîner des modèles d'apprentissage automatique tout en préservant la confidentialité des données est un objectif de longue date [12]. Dans le cadre de l'apprentissage automatique traditionnel, les systèmes embarqués envoient, via un réseau de communication, leurs données brutes à un serveur puissant, qui effectue l'apprentissage du modèle et le renvoie. Toutefois, ce processus soulève des problèmes de confidentialité, tels que l'interception des données pendant la communication et l'accès non autorisé aux données des utilisateurs par le propriétaire du serveur ou un tiers. Dans l'Apprentissage Fédéré (AF) [11] standard, le serveur envoie un modèle à un groupe de clients, qui l'entraînent sur leurs données locales, puis renvoient leurs paramètres mis à jour au serveur pour agrégation. En inversant le processus d'entraînement de cette manière, l'AF essaye de mieux garantir la confidentialité des données de l'utilisateur puisque les données originelles ne quittent jamais les terminaux des clients. Ces terminaux embarqués tels que les dispositifs *IoT*, les *smartphones* et les drones sont bien adaptés aux applications de l'AF en raison de leur proximité avec les données et les applications du monde réel [14]. Cependant, beaucoup de ces appareils ont des ressources de calcul limitées et des techniques de co-conception [14] sont continuellement explorées pour faire correspondre les algorithmes aux contraintes matérielles. Parmi les sujets de recherche émergents pour l'AF au niveau *edge*, le domaine de la compression de réseaux de neurones est un moyen prometteur de s'attaquer aux contraintes des dispositifs exploitant l'AF [9].

Des travaux précédents ont considéré l'application de la quantification [16] et de l'élagage [8] à l'AF pour améliorer les performances du modèle. Notre approche est d'appliquer la quantification et l'élagage à un scénario d'AF typique afin de mesurer leurs impacts sur le compromis entre compression et précision du modèle final, dans le but d'évaluer leur efficacité à réduire les communications tout en conservant de bonnes performances.

2 Contexte

2.1 Apprentissage fédéré

L'Apprentissage Fédéré (AF) [11] est un *framework* distribué qui permet l'entraînement collaboratif de modèles d'apprentissage automatique ou *Machine Learning* (ML) sur plusieurs dispositifs, appelés clients, via un coordinateur central, généralement un serveur avec de grandes ressources de calcul. Les clients sont couramment des dispositifs embarqués, comme les *smartphones*. Contrairement au ML traditionnel, chaque client entraîne son propre modèle de ML local et ne partage que les paramètres du modèle ou ses gradients, avec le serveur, ce qui permet à plusieurs clients d'entraîner conjointement un modèle sans partager leurs données. À chaque cycle d'entraînement fédéré, également appelé « *round* », le serveur envoie le modèle à un sous-ensemble de clients, qui l'entraînent localement et renvoient les paramètres ou gradients mis à jour. L'étape finale consiste à agréger, côté serveur, les paramètres des clients sélectionnés pour créer un modèle global, qui peut bénéficier des connaissances acquises par chaque client. Chaque *round* implique, le téléchargement du modèle et plusieurs itérations d'entraînement au niveau du client. Ainsi, même sans partager les données privées, les clients peut bénéficient d'un modèle qui a été amélioré par des jeux de données plus diversifiés.

2.2 Compression de modèle

La compression de modèle est une solution largement adoptée [2] pour réduire les besoins d'un modèle en termes de calculs et de mémoire. Parmi les techniques de compression existantes, la quantification et l'élagage ont été mis en œuvre pour réduire la complexité de l'inférence et de l'entraînement des réseaux de neurones dans les dispositifs embarqués [5, 10]. Ces deux techniques sont prises en compte dans cette étude.

Les modèles de réseaux de neurones sont généralement construits à l'aide de nombres à virgule flottante 32 bits (FP32), qui sont plus coûteux en termes de calcul, de mémoire et d'énergie que les nombres entiers [6]. La quantification est un sujet d'intérêt [2] dans les réseaux de neurones car elle permet des formats plus petits et/ou plus simples, qu'il s'agisse de nombres à virgule flottantes sur un plus petit nombre de bits, de nombre entiers, ou encore de formats hybrides, ce qui permet une diminution significative de la complexité de calcul et des besoins en mémoire, mais au prix d'une potentielle pénalité sur la précision de classification. Dans notre approche, nous avons utilisé l'entraînement tenant compte de la quantification (« *Quantization-Aware-Training* », QAT) qui permet au réseau d'apprendre aussi le bruit introduit par le processus de quantification [10, Appendix C].

L'élagage vise quant à lui à réduire la complexité d'un modèle en supprimant les parties redondantes ou inutiles d'une architecture. Un modèle large et profond donne des résultats à l'état de l'art, mais la contribution de chacun de ses éléments à la performance du réseau total n'est pas homogène. Ainsi, en observant chaque élément architectural du réseau, il est possible d'éliminer ceux qui ont peu d'impact. Il existe deux approches possibles de l'élagage dans le contexte des réseaux de neurones. La première consiste à élaguer individuellement les poids de plus faible magnitude, ce qui est communément appelé élagage non structuré. La seconde approche consiste à élaguer des structures entières au sein du réseau, telles que les noyaux, les filtres ou les couches de convolution, qui ne contribuent pas de manière significative aux performances du réseau. Cette approche est dénommée « élagage structuré ».

La quantification des poids permet de réduire considérablement la taille des messages tout en maintenant la performance du modèle. D'un autre côté, l'élagage non structuré peut également offrir des avantages en termes de compression grâce à l'utilisation de techniques de codage entropique, telles que le codage de Huffman [4], en exploitant les paramètres épars.

3 Méthodologie

Nous avons mené des expériences visant à étudier le comportement de la quantification et de l'élagage avec des données non-IID (non indépendantes et identiquement distribuées), tout en tenant compte de l'impact sur la compression des messages. Nous avons appliqué différents niveaux de quantification et d'élagage dans le cas d'une distribution fixe des données d'un client non-IID pour détecter les compromis entre la compression et les mécanismes d'entraînement de l'AF selon le dispositif expérimental illustré à la Fig. 1. En partant du pipeline de l'AF standard, nous introduisons deux modifications : QAT et élagage. Dans l'expérience de quantification (à gauche), seuls les clients doivent être modifiés pour inclure la méthode d'apprentissage QAT. Dans l'expérience d'élagage (à droite), les clients et le serveur appliquent l'élagage et la compression. Notre code est publiquement accessible[1].

Nous avons utilisé le *framework* de Flower [1] pour simuler 10 clients d'AF. À chaque *round*, 40 % des clients sont sélectionnés pour le processus d'entraînement, et 100 rounds ont été réalisés pour étudier l'évolution de la précision du modèle du serveur. Chaque client a utilisé SGD avec *momentum* comme optimiseur. Pour la simplicité, les hyperparamètres sont conformes à [15]. Le serveur utilise FedAvg comme stratégie d'agrégation. Les exemples d'entraînement sont répartis sur les clients avec une Allocation de Dirichlet Latente (ADL) [7] sur l'ensemble d'entraînement. Suivant [15], nous avons remplacé la couche BatchNorm par une couche GroupNorm [17]. Nous avons choisi de travailler avec des niveaux de quantification de 1 bit, 4 bits et 8 bits, utilisant Binary Connect [3] pour les réseaux binaires et le *framework* Brevitas [13] pour 4 et 8 bits avec le schéma de quantification par défaut. Les poids sont quantifiés en entiers de 4 et 8 bits et la mise à l'échelle de la QAT est calculée par couche. Enfin, pour l'élagage, nous avons opté pour une méthode basée sur la magnitude non structurée, dans laquelle chaque client élaguait les θ % des plus petits poids à la fin de chaque round. Les poids élagués sont simplement des poids à valeur nulle et sont considérés comme des paramètres normaux lors de l'agrégation. Le serveur procède également à l'élagage en utilisant le même ratio θ % que les clients à chaque round, pour maintenir la cohérence entre les flux ascendants et descendants.

4 Résultats expérimentaux

Le modèle, ResNet12, comporte 780K paramètres pouvant être entraînés, qui sont communiqués entre le serveur et le client à chaque *round*, occupant 2.97 MiO, avec le format FP32. Si l'on prend l'exemple de la quantification des poids sur 1, 4 et 8 bits, le message est réduit à 0.10, 0.38 et 0.75 MiO, respectivement. Cela se traduit par une compression de 4-32 fois, si l'on considère uniquement la quantification, qui conduit à des réductions supplémentaires car les paramètres sont envoyés deux fois pour chaque *round* et le nombre de *rounds* peut aller jusqu'à quelques milliers [15].

En examinant Fig. 2, on constate que le temps de convergence, c'est-à-dire le nombre de *rounds* nécessaires pour atteindre une précision maximale, n'est pas le même d'une expérience à l'autre, car il dépend également du niveau de quantification. Localement, chaque client entraîne le modèle pendant un certain nombre d'époques, avant de renvoyer les paramètres au serveur. Comme expliqué dans [11, 15], les époques locales sont un hyperparamètre important à régler et contribuent à l'effet de dérive du client [9]. Dans ces figures, outre le fait que l'entraînement avec 4 et 8 bits nous permet d'obtenir une précision comparable à la référence, il révèle également un compromis entre la communication et le calcul. Afin d'atteindre une précision similaire d'environ 75 %, la Fig. 2, il est nécessaire d'effectuer 40 *rounds* de communication et 40 époques totales lorsqu'on utilise une époque locale, tandis que dans le cas de 10 époques locales, il faut effectuer 100 époques totales et 10 *rounds*, un rapport calcul et communication de 4. Toutefois, dans le cas d'un bit, le passage du nombre d'époques par *round* sur le client de 1 à 10 augmente

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

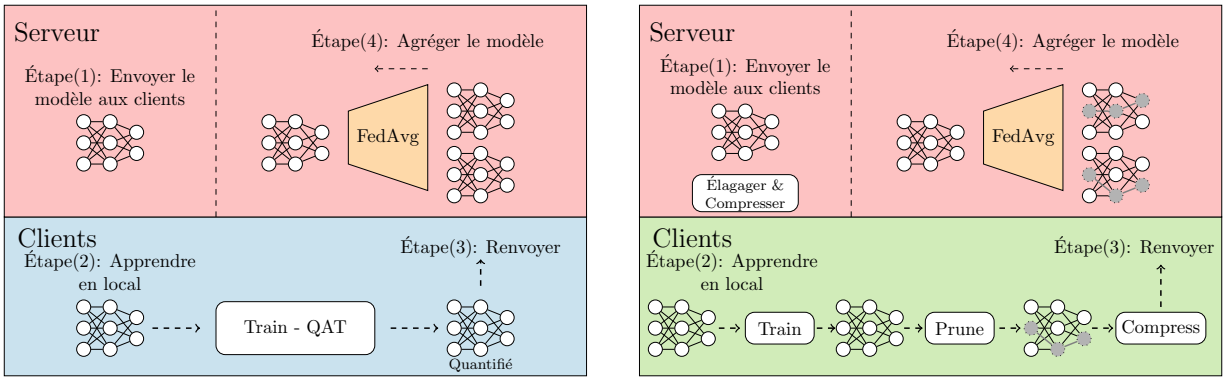

FIGURE 1 : Modes opératoires d'apprentissage de notre étude de compression en apprentissage fédéré : sur la gauche sont représentées les étapes de quantification de modèle et sur la droite celles d'élagage.

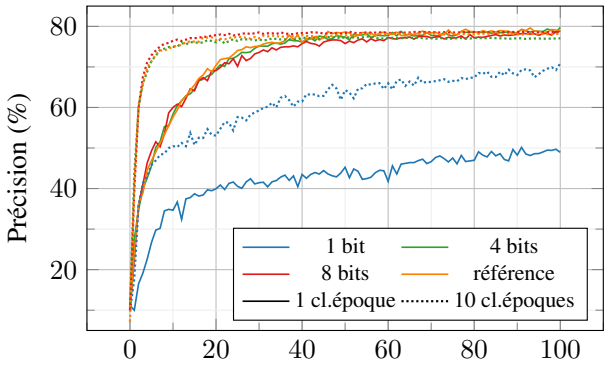

FIGURE 2 : Comparaison de l'évolution de la précision entre la référence (FP 32-bits), 1-, 4- et 8-bits, pour 1 et 10 époques locales.

considérablement la précision, qui passe de 48.8 % à 70.9 %, le nombre total d'époques passant de 100 à 1000, mais avec le même coût de communication. Le nombre d'époques par *round* sur le client a un impact sur les résultats de l'élagage dans Fig 3. Passer plus de temps dans chaque client contribue à un modèle plus robuste, ce qui permet des communications de données plus éparses tout en conservant approximativement la même précision, même si cela augmente le nombre total d'époques.

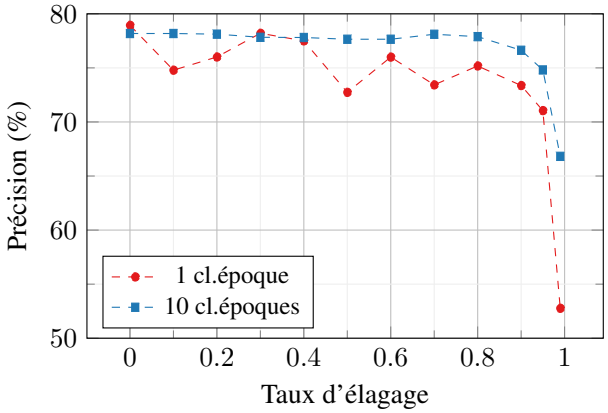

FIGURE 3 : Effet de l'élagage sur la précision en fonction du taux d'élagage, où le taux représente le pourcentage de du nombre de paramètres élagués, pour 1 et 10 époques par *round*.

TABLE 1 : Résumé des tailles des messages et précisions pour CIFAR-10

Compression	**Précision (%)**		**Message (MiO)**
	1 Epoque local	10 Epoques locales	
Référence	78.94	78.18	2.97
Élagage			
10 %	74.79	78.18	2.33
20 %	76.01	78.12	2.09
30 %	78.20	77.83	1.84
40 %	77.50	77.81	1.59
50 %	72.74	77.65	1.33
60 %	76.00	77.65	1.06
70 %	73.43	78.11	0.81
80 %	75.18	77.89	0.56
90 %	73.37	76.63	0.31
95 %	71.05	74.81	0.20
99 %	52.77	66.82	0.12
Quantification			
1-bit	48.93	70.89	0.10
4-bits	79.74	77.04	0.38
8-bits	78.80	78.58	0.75

Le tableau 1 résume la taille d'un message échangé entre client et serveur pour l'entraînement en AF. Pour la quantification, la taille du message dépend uniquement des poids quantifiés, car le serveur connaît la quantification du client. Pour l'élagage, nous avons compressé le modèle à l'aide de l'algorithme ZIP. Le tableau 1 montre également que même des approches simples peuvent être utilisées pour compresser un réseau, ce qui représente des économies de 2 à 4 fois en termes de bande passante sans nuire significativement la précision.

5 Discussions et travaux futurs

Le défi consiste à obtenir une compression élevée avec un faible nombre d'époques d'apprentissage, tout en conservant la précision du modèle. Les méthodes explorées ici sont orthogonales à la fonction d'agrégation utilisée ou à d'autres optimisations globales/locales.

La compression des modèles est nécessaire non seulement

pour réduire les coûts de communication, mais aussi en raison des capacités limitées de ces dispositifs en calcul et en mémoire. Cependant, l'intégration de l'élagage et de la quantification dans l'apprentissage peut nuire aux performances du client. Les outils utilisés dans cette étude, Brevitas et l'élagage par magnitude, ont des limites. Brevitas n'est pas conçu pour un apprentissage efficace, doublant le temps d'apprentissage, et l'élagage de la magnitude trie et agit sur tous les paramètres, ce qui est coûteux pour les modèles de grande taille. Des travaux récents ont proposé des méthodes plus efficaces pour la compression des réseaux de neurones, telles que l'entraînement quantifié efficace et l'élagage, mais leur viabilité en AF reste à prouver. En outre, dans un scénario réel, le paradigme classique de l'AF, qui suppose que tous les clients et le serveur partagent la même architecture de réseau de neurones, n'est pas satisfaisante. Cette approche peut limiter la participation des clients qui peuvent avoir des architectures de modèle différentes ou dont les appareils ont des capacités de calcul et de mémoire limitées et différentes les unes des autres. Il est donc important d'envisager le paradigme de l'hétérogénéité des modèles d'AF. Plusieurs travaux récents ont exploré cette direction, comme FedET [8], qui étudie comment apprendre efficacement des modèles avec des architectures et des complexités différentes dans un cadre d'AF. On peut en outre supposer que la combinaison de la quantification et de l'élagage pourrait améliorer les performances que nous présentons, même si les résultats que nous montrons sont suffisants pour justifier la pertinence des deux indépendamment.

L'AF introduit un nouveau paradigme dans lequel les modèles doivent être entraînés de manière distribuée, tout en tenant compte des contraintes matérielles qui peuvent avoir un impact à la fois sur le modèle global et sur les modèles des clients. Cela pose plusieurs défis en termes d'entraînement, notamment la nécessité d'optimiser les modèles pour des données non-IID et pour réduire les coûts de communications. En outre, le framework d l'AF peut impliquer des jeux de données qui évoluent dans le temps, il est donc important de prendre en compte les possibles implications de l'apprentissage continu.

6 Conclusion

L'apprentissage fédéré est un nouveau paradigme dans lequel les modèles doivent être entraînés de manière distribuée, tout en tenant compte des contraintes matérielles qui peuvent avoir un impact à la fois sur le modèle global et ceux des clients. Pour adapter les modèles à des systèmes embarqués, il est possible d'avoir recours à des techniques de compression telles que l'élagage et la quantification. Ces techniques, en réduisant la taille des modèles permettent également de réduire les coûts de communication entre clients et serveurs, au prix toutefois d'une réduction de la précision. Dans cet article, nous décrivons un processus d'élagage et de quantification, et menons des expériences pour mesurer leur impact sur le compromis entre coûts de communication précision des modèles.

Remerciements

Ce travail est soutenu par le programme *Futur et Ruptures* financé par l'IMT et l'Institut Carnot TSN, et par le GdR ISIS.

Références

[1] Daniel J et al BEUTEL : Flower : A friendly federated learning research framework. *arXiv :2007.14390*, 2020.

[2] Yu CHENG et et AL : A survey of model compression and acceleration for deep neural networks. *arXiv :1710.09282*, 2017.

[3] Matthieu et al COURBARIAUX : Binaryconnect : Training deep neural networks with binary weights during propagations. *Advances in neural information processing systems*, 28, 2015.

[4] Song et al HAN : Deep compression : Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv :1510.00149*, 2015.

[5] Torsten et al HOEFLER : Sparsity in deep learning : Pruning and growth for efficient inference and training in neural networks. *J. Mach. Learn. Res.*, 22(241):1–124, 2021.

[6] Mark 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)*, pages 10–14, 2014.

[7] Tzu-Ming Harry et al HSU : Measuring the effects of non-identical data distribution for federated visual classification. *arXiv :1909.06335*, 2019.

[8] Yuang aet al JIANG : Model pruning enables efficient federated learning on edge devices. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[9] Peter et al KAIROUZ : Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

[10] Ji et al LIN : On-device training under 256kb memory. *arXiv :2206.15472*, 2022.

[11] Brendan et al MCMAHAN : Communication-efficient learning of deep networks from decentralized data. *In Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[12] Fatemehsadat et al MIRESHGHALLAH : Privacy in deep learning : A survey. *arXiv :2004.12254*, 2020.

[13] Alessandro PAPPALARDO : Xilinx/brevitas, 2021.

[14] Partha Pratim RAY : A review on tinyml : State-of-the-art and prospects. *Journal of King Saud University-Computer and Information Sciences*, 2021.

[15] Sashank et al REDDI : Adaptive federated optimization. *arXiv :2003.00295*, 2020.

[16] Nicola et al TONELLOTTO : Neural network quantization in federated learning at the edge. *Information Sciences*, 575:417–436, 2021.

[17] Yuxin WU et Kaiming HE : Group normalization. *In Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.