Communication-Efficient Federated Learning through Adaptive Weight Clustering and Server-Side Distillation

Vasileios Tsouvalas[1]   **Aaqib Saeed**[1]   **Tanir Ozcelebi**[1]   **Nirvana Meratnia**[1]

[1]Eindhoven University of Technology, The Netherlands

Abstract

Federated Learning (FL) is a promising technique for the collaborative training of deep neural networks across multiple devices while preserving data privacy. Despite its potential benefits, FL is hindered by excessive communication costs due to repeated server-client communication during training. To address this challenge, model compression techniques, such as sparsification and weight clustering are applied, which often require modifying the underlying model aggregation schemes or involve cumbersome hyperparameter tuning, with the latter not only adjusts the model's compression rate but also limits model's potential for continuous improvement over growing data. In this paper, we propose `FedCompress`[1], a novel approach that combines dynamic weight clustering and server-side knowledge distillation to reduce communication costs while learning highly generalizable models. Through a comprehensive evaluation on diverse public datasets, we demonstrate the efficacy of our approach compared to baselines in terms of communication costs and inference speed.

Federated learning (FL) enables collaborative training of neural network models directly on edge devices (referred to as clients), preserving on-device data locally Konečný et al. (2016). FL is composed of multiple federated rounds, which involve server-to-client model updates dispatch, local training by clients, and server-side aggregation (e.g., *FedAvg* McMahan et al. (2017)) of clients' model updates, iteratively performed until model convergence. Despite its appealing properties for users' privacy, FL requires constant model transportation between server and clients, which poses a challenge in terms of communication efficiency. This becomes even more critical when the clients are resource-constrained edge devices with limited computational capabilities and strict energy constraints.

To address the communication overhead in FL, recent studies have explored model compression schemes on the exchanged model updates to minimize the communication overhead. Sparsification Stich et al. (2018) involves discarding network segments to reduce the overall model's complexity based on a threshold value. FedSparsify Stripelis et al. (2022) utilizes magnitude pruning with a gradually increasing threshold during training to learn a highly-sparse model in a communication-efficient FL scheme. Alternatively, weight clustering Han et al. (2015) converts the weight matrices elements into a discrete set of values (clusters) to achieve high model compression ratio. MUCSC Cui et al. (2021) utilizes layer-wise weight clustering using a fixed number of clusters to communicate compressed model updates from clients to server. Likewise, FedZip Malekijoo et al. (2021) employs a sequence of pruning and weight clustering to further improve the compression ratio. Apart from these model compression schemes, knowledge distillation Wu et al. (2022) and sub-model training Rabbani et al. (2023); Niu et al. (2023) have being explored to reduce communication costs. However, the aforementioned techniques require modifications to the underlying model aggregation algorithm and solely focus in optimizing the client-to-server (upstream) communication route. Furthermore, existing weight clustering schemes in FL rely on a fixed set of clusters, limiting model's potential for continuous improvement over growing data.

We propose `FedCompress` (*Federated Learning with Dynamic Weight Clustering for Model Compression*) to achieve significant communication reductions in the bidirectional communication route during FL training, while
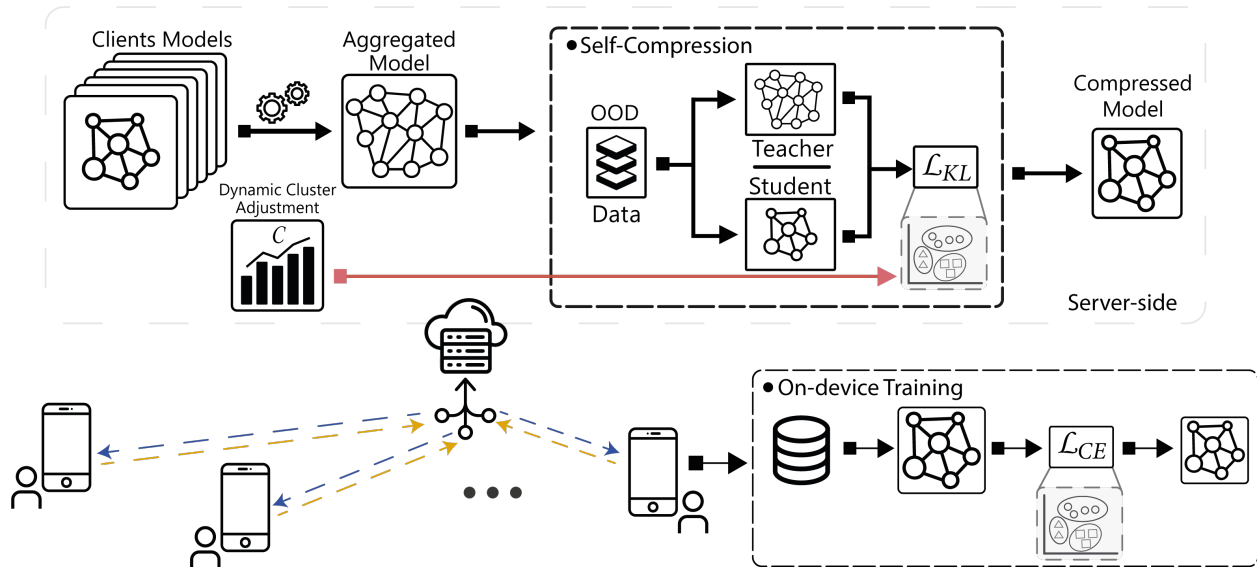
---

[1]https://github.com/FederatedML/FedCompress

Figure 1: Illustration of FedCompress for communication-efficient FL. A dual-stage compression scheme is proposed: (i) weight clustering across clients during on-device training, and (ii) self-compression on server-side combining weight clustering with knowledge distillation on out-of-distribution data.

maintaining the representational power of models and achieving highly compressed models. Naively applying weight clustering during local updates in FL yields limited compression benefits for the aggregated model, as clients' models form diverse clusters during compression due to the statistical heterogeneity among clients. Consequently, the centroid-based structure of the aggregated model weights is compromised, once the server-side aggregation is applied Briggs et al. (2020). To overcome this issue with no modifications in the underlying aggregation strategy, we apply the knowledge distillation Hinton et al. (2015) scheme using out-of-distribution data after model aggregation at the server. In this way, we perform self-compression on the aggregated model, enforcing the centroid-like structure and reducing communication costs in the downstream communication channel for the next federated round.

Aside from the server-side compression, we balance the trade-off between model performance and communication efficiency by finding a suitable number of clusters during the FL process. To this end, we start with a small number of clusters for each layer and dynamically adjust them as federated rounds progress by monitoring the representational power of clients' models. In particular, we propose a representation quality score, which is computed locally at each client using a small unlabelled set of clients' available data. Concisely, the main contributions of our work are as follows:

- We propose `FedCompress`, a communication efficient FL approach that combines weight clustering and knowledge distillation on out-of-distribution data to achieve highly compressed models, with no modifications to the underlying weights' aggregation algorithm.

- We introduce a representation quality score, locally computed based on clients' unlabelled data, to dynamically adjust the number of clusters utilized during weight clustering.

- We demonstrate that proposed method is highly effective for learning generalizable compressed federated models on diverse public datasets from both vision and audio domains, namely CIFAR-10 Krizhevsky et al. (2009), CIFAR-100 Krizhevsky et al. (2009), PathMNIST Yang et al. (2021), Speech-Commands Warden (2018) and VoxForge MacLean (2018).

- Our evaluation of `FedCompress` shows on average a 4.5-fold reduction in communication costs during training and a 1.13-fold speedup during inference on edge accelerator devices when compared to *FedAvg* across considered datasets.

1 Methodology

1.1 Problem Formulation

We focus on the problem of federated model compression to overcome the high communication-burden in FL and offer notable inference speedup on edge devices as a by-product. Formally, we assume that each of the K clients has a labeled and a small validation (unlabeled) dataset, denoted by $\mathcal{D}_l^k = \{x_i, y_i\}_{i=1}^{N_l^k}$, and $\mathcal{D}_u^k = \{x_i\}_{i=1}^{N_u^k}$, respectively. Furthermore, server possesses an OOD dataset, $\mathcal{S} = \{x_i\}_{i=1}^{N_s}$, which can have no statistical similarity with clients locally stored data. We aim to learn a compressed global model, $p_{\tilde{\theta}}$ without clients sharing locally stored data, $\mathcal{D}_l$ and $\mathcal{D}_u$, with the server and reduce the communication burden during FL training phase. Here, we denote with p_θ a neural network with weights θ, while $\tilde{p}_\theta^k$ and $\tilde{\theta}$ refer to the weight-clustered (compressed) versions of p_θ and θ, respectively.

1.2 Federated Model Compression

We propose `FedCompress`, a two-stage compression scheme for bi-directional communication reduction during training in FL. During local model training, we simultaneously train and compress clients' models using weight-clustering to reduce the upstream communication costs. To maintain a highly compressed model once server-side model aggregation is complete, we employ a novel model compression scheme that utilizes OOD data to minimize the downstream communication burden, while maintaining high model performance. Furthermore, To strike a balance between model performance and communication efficiency trade-off, we propose a dynamically adaptive weight-clustering approach to monitor and update the number of clusters based on the representational power of clients' models, allowing `FedCompress` to adapt to the underlying task's complexity.

Client-side Model Compression: During the local model update step, we initially apply standard cross-entropy to each of the k client's labeled datasets, $\mathcal{D}_l^k$, while simultaneously enforcing the weights θ to cluster around a set of C learnable centroids, μ. Specifically, each client's minimization objective is defined as follows:

$$\min_\theta \mathcal{L}_k(\theta_k) = \mathcal{L}_{ce}(p_{\theta_k}(\mathcal{D}_l^k)) + \beta \cdot \mathcal{L}_{wc}(\theta_k, \mu, C) \tag{1}$$

Here, $\mathcal{L}_{ce}$ is the cross-entropy loss function for the model p_θ on the labeled dataset $\mathcal{D}_l^k$. We use β to control the relative impact of $\mathcal{L}_{ce}$ and $\mathcal{L}_{wc}$. As the initial centroids are important to maintain model's high representational power, we allow for a few training rounds using $\mathcal{L}_{ce}$ before introducing $\mathcal{L}_{wc}$. In essence, we start each local FL training step with $\beta=0$ for a few epochs and afterwards setting $\beta=1$.

Self-Compression on Server: Once the server has constructed a new global model from the received model updates, the centroid-shape structure of model weights is compromised, making it challenging to maintain a compressed model for downstream communication. To solve this problem, we propose a self-compression mechanism that combines weight clustering and knowledge distillation on OOD data, $\mathcal{S}$, at the server side. This mechanism involves training a compressed version of the original global model (i.e., the teacher), which acts as a student that aims to mimic the behavior of its teacher on the OOD data.

In this way, we enforce the model weights to cluster around a set of C learnable centroids, similar to the client-side compression, while recovering any performance degradation due to weight-clustering. As a loss function to this teacher-student architecture, we utilize the Kullback-Leibler divergence (KLD) loss, which aims to match the output distributions of the models. Specifically, the objective function of the proposed server-side self-compression approach is as follows:

$$\min_\theta \mathcal{L}_s(\theta) = \mathcal{L}_{kl} \left(p_{\theta_T} \,||\, p_\theta \right) + \beta \cdot \mathcal{L}_{wc}(\theta, \mu, C)$$

$$= \lambda^2 \cdot \sum_{x \in \mathcal{S}} p_{\theta_T}^\lambda(x) \log \frac{p_{\theta_T}^\lambda(x)}{p_\theta^\lambda(x)} + \beta \cdot \sum_{j=1}^{C} \sum_{i=1}^{N_s} u_{ij}||\theta_i - \mu_j||^2 \qquad (2)$$

where λ is a temperature scalar, $\mathcal{L}_{wc}$ is the cross-entropy loss, $\mathcal{L}_{kl}$ is the KLD loss being computed using $p_{\theta_T}^\lambda$ and p_θ^λ the λ-scaled logits of teacher and student models, respectively.

One may note that no labels are required to enforce the alignment of distributions, thus any unlabelled dataset available on the server can be utilized to perform the self-compression. Furthermore, with no modifications required to the underlying model aggregation mechanism (e.g., *FedAvg*), our method provides a readily usable solution to any existing FL systems to reduce the downstream communication costs.

Dynamic Weight-Clustering: While model compression via weight clustering can result in highly accurate compressed models, its performance is directly affected by selecting a proper number of clusters, C. Especially, in FL, where clients have heterogeneous data distributions, a suitable number of clusters can vary significantly based on the heterogeneity of clients and their local data distributions. We propose dynamically adjusting the number of clusters based on clients models' representational power. In particular, we assess model's performance using embeddings from the penultimate layer of the model, where we use the rank of embeddings as a proxy of their generalization quality Roy and Vetterli (2007). This representation quality score is computed locally on client's unlabeled dataset $\mathcal{D}_u^k$.

Formally, with the completion of the local training step on k-th client, we extract the embeddings $\mathcal{Z}^k$ from $\mathcal{D}_u^k$ using client's model p_{θ_k}, and afterwards compute the score, $\mathcal{E}_k$ using $\exp(-\sum_{j=1}^{m_{\mathcal{Z}^k}} r_j \log r_j)$, where r_j denoting the ranking of j-th singular value of $\mathcal{Z}^k$ ($\frac{\sigma_j}{|\sigma_{\mathcal{Z}^k}|_1}$) and $m_{\mathcal{Z}^k}$ denotes the minimum dimension of embeddings. To ensure numerical stability, we add a small constant equal to $1e^{-7}$ in the computation of r_j. During the server-side model aggregation step in each federated round, we compute the weighted-averaged representation quality score $\mathcal{E}$ of participating clients models, similar to *FedAvg*.

To ensure that the model is compressed efficiently, we start with a small value for C (i.e., minimum number of clusters - C_{min}), incrementing it when the moving average of $\mathcal{E}$ over a window W shows no improvement in the previous P rounds. We fix $W=3$ and $P=3$, which we determine to be working well during our initial exploration. Furthermore, our approach allows for a maximum communication budget to be specified prior to FL training (e.g. based on an energy consumption profile) to update C between two boundary values, $[C_{min}, C_{max}]$. Further details and an overview of our proposed `FedCompress` approach for communication-efficient FL can be found in Algorithm 1.

2 Evaluation

Datasets: We use publicly available datasets from both the vision and audio domains with their standard training/test splits. Specifically, we use the CIFAR-10/100 Krizhevsky et al. (2009) and PathMNIST Yang et al. (2021) datasets, where the tasks of interests are object detection and pathology reporting, respectively. Likewise, from the audio domain, we use SpeechCommands Warden (2018) for keyword spotting (12 classes in total), and VoxForge MacLean (2018) for language identification. As OOD datasets for self-compression on server, we use StyleGAN (Oriented) Baradad Jurjo et al. (2021) and Librispeech Panayotov et al. (2015) for our vision and audio recognition tasks, respectively. However, we note that augmented patches (or segments in case of audio) from a single image can also be used as OOD data Asano and Saeed (2023).

Experimental Setup: We utilize ResNet-20 He et al. (2016) for the vision domain, while we choose MobileNet Howard et al. (2017) for the considered audio recognition task. These models were chosen based on their performance, suitability for on-device learning, where computational resources are limited compared to centralized settings, and extensive validation in previous research Tsouvalas et al. (2022). To simulate a federated environment, we use Flower Beutel et al. (2020) with *FedAvg* McMahan et al. (2017) to construct

Table 1: Experimental results reporting Communication Compression Reduction (CCR), Model Compression Ratio (MCR) and accuracy difference δ-Acc versus standard *FedAvg* for FedZip Malekijoo et al. (2021) and `FedCompress` (with and without Self-Compression on Server - SCS). Results are reported across five datasets, while CCR and MCR are indicating an n-fold reduction from the standard *FedAvg* results. Federated parameters are set to R=20, M=20, E_c=10, E_s=10, and σ=25%.

Dataset	FedAvg Accuracy	FedZip Malekijoo et al. (2021)			FedCompress (*w/o SCS*)			FedCompress		
		δ-Acc	CCR	MCR	δ-Acc	CCR	MCR	δ-Acc	CCR	MCR
CIFAR-10	86.26	-1.89	1.91	2.08	-1.47	1.02	1.77	-1.83	4.53	5.18
CIFAR-100	60.68	-2.57	1.94	2.11	-2.67	1.02	1.62	-1.88	3.80	3.93
PathMNIST	88.22	-3.04	1.92	2.10	-3.57	1.06	1.82	-1.72	4.79	5.27
SpeechCommands	95.75	-0.82	1.66	1.88	-0.72	1.06	1.72	-0.42	5.04	5.09
VoxForge	81.05	-1.04	1.69	1.91	0.75	1.11	1.81	-0.31	5.41	5.64

the global model from clients' local updates. We control the federated setting in our experiments with the following parameters: 1) number of clients - M=20, 2) number of rounds - R=20, 3) local train epochs - E_c=10, server self-compression training epochs - E_s=10, 4) data distribution variance across clients - σ=25%. We randomly partitioned the datasets across the available clients in a non-overlapping fashion.

From the related approaches in the literature, we performed experiments using FedZip Malekijoo et al. (2021) with number of clusters fixed to 15 (which we find to work well for the considered tasks after preliminary experimentation), while compared both `FedCompress` and FedZip performances in terms of test set accuracy, communication-cost reduction (CCR), and model compression ratio (MCR) with respect to the standard *FedAvg*. For a rigorous review, we manage any randomness during data partitioning and training procedures with a seed value and performed two distinct trials, reporting the average accuracy on test sets.

Results: In Table 1, we report our findings across all datasets and compare `FedCompress` performance with the considered baselines. As indicated from δ-Acc (accuracy versus standard *FedAvg*) columns in Table 1, `FedCompress` outperforms FedZip across all considered datasets and yields compressed federated models with comparable performance to *FedAvg*, while achieving significant reductions in communication costs. On audio-based tasks, where the MobileNet model was utilized, we observe over a 5-fold reduction in communication costs (CCR), while keeping models' accuracy within 0.5% of the standard FL process. In the vision domain, our approach remains equally effective, with a CCR of 4.28, while suffering an accuracy drop of approximately 2.14% across all image dataset.
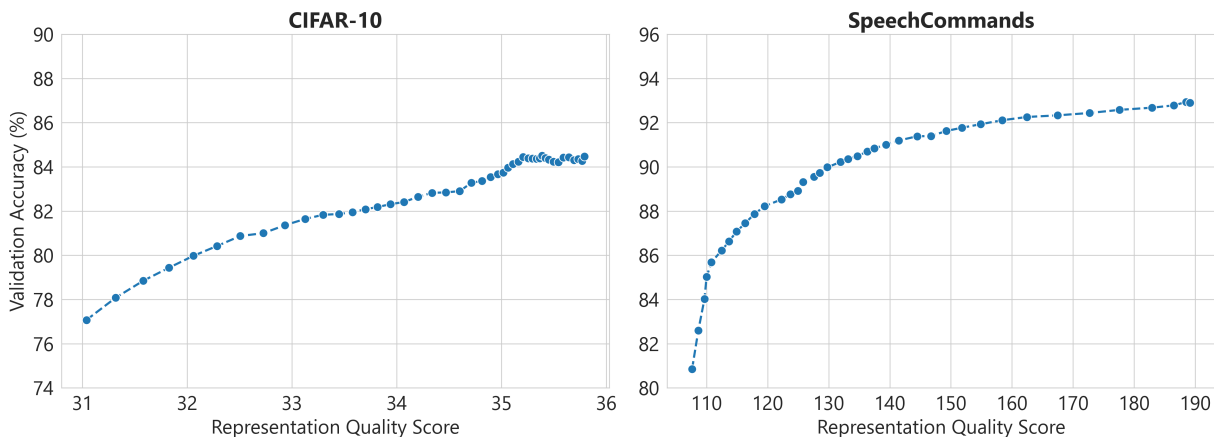

Figure 2: Relationship between mean representation quality score and mean validation accuracy across clients during FL training for `FedCompress` on CIFAR-10 and SpeechCommmands. Strong positive correlation is observed, indicating that the representation quality score is a useful indicator of the clients models' representational power.

Overall, an average 4.5-fold CCR is achieved throughout all five datasets, suggesting that `FedCompress` can effectively reduce the communication costs in FL without sacrificing models' accuracy. These results indicate

Table 2: Inference time acceleration of diverse edge devices for ResNet-20 and MobileNet on CIFAR-10 and SpeechCommands datasets, respectively. The reported inference time acceleration is achieved by comparing to FL models using *FedAvg*.

Model	Device	float32	uint8 (Quantized)
ResNet-20	Pixel 6	×1.103	×1.165
	Jetson Nano	×1.127	×1.169
	Coral TPU	×1.113	×1.191
MobileNet	Pixel 6	× 1.114	×1.248
	Jetson Nano	×1.137	×1.161
	Coral TPU	×1.152	×1.194

the potential for significant energy savings and reduced communication bandwidth requirements in FL, both crucial for resource-constrained devices and Internet of Things (IoT). Apart from the reduction in communication costs, `FedCompress` demonstrates a significant reduction in resulting model size. We observe an average MCR of 4.14 across all datasets, indicating that our approach can effectively compress models without sacrificing their accuracy. The highly compressed models offer additional benefits to edge devices, such as lower memory requirements and reduced energy consumption during training and inference. This, in turn, allows for more complex models to be deployed on edge devices with limited resources.

To validate the utilization of our representation quality score as an indicator of models' representation power, we compute the score and the validation accuracy across clients in each federated round on CIFAR-10 and SpeechCommands and compared their progression. Figure 2 shows a strong correlation between the two metrics, indicating that the models' representation quality metric is a valuable alternative to the validation accuracy, while it can be efficiently computed from clients models' embeddings with no need for having labeled data. Since unlabeled data are often readily available on edge devices, our score can effectively act as a proxy to dynamically set the number of clusters during training, providing a lightweight and fast alternative for measuring models' representational power.

We also conducted experiments on various edge devices to evaluate the impact of our compression approach on inference time. Table 2 shows that our approach can accelerate the inference time on all these devices, while maintaining comparable accuracy to *FedAvg* as shown in Table 1. Notably, `FedCompress` models demonstrate an acceleration of up to $1.15\times$ across the edge devices, while quantizing the models to uint8 achieves an inference speedup up to $1.24\times$. Consequently, `FedCompress` provides the ability to achieve faster inference times with minimal effect on federated models' accuracy, providing significant benefits for resource-constrained edge devices, where low power consumption and reduced inference time are crucial.

3 Conclusion

We presented `FedCompress`, a communication-efficient FL approach based on model-compression via weight clustering and knowledge distillation. It can be easily integrated with existing FL frameworks, requiring no modifications to the FL aggregation strategy. Our experiments across multiple datasets showed that our approach can achieve significant reductions in communication costs and model sizes, while maintaining comparable accuracy to the standard FL process. Moreover, we have shown that our proposed embeddings-based representation quality score can effectively act as a proxy for models' representational power, allowing for a dynamic adjustment of the number of clusters during training based on the underlying task's complexity.

Acknowledgement

References

Yuki M Asano and Aaqib Saeed. The augmented image prior: Distilling 1000 classes by extrapolating from a single image. In *The Eleventh International Conference on Learning Representations*, 2023.

Manel Baradad Jurjo, Jonas Wulff, Tongzhou Wang, Phillip Isola, and Antonio Torralba. Learning to see by looking at noise. *Advances in Neural Information Processing Systems*, 34:2556–2569, 2021.

Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Titouan Parcollet, and Nicholas D Lane. Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390*, 2020.

Christopher Briggs, Zhong Fan, and Peter Andras. Federated learning with hierarchical clustering of local updates to improve training on non-iid data. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE, 2020.

Laizhong Cui, Xiaoxin Su, Yipeng Zhou, and Yi Pan. Slashing communication traffic in federated learning by transmitting clustered model updates. *IEEE Journal on Selected Areas in Communications*, 39(8):2572–2589, 2021.

Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. 2015. doi: 10.48550/ARXIV.1510.00149. URL https://arxiv.org/abs/1510.00149.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtarik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. In *NIPS Workshop on Private Multi-Party Machine Learning*, 2016. URL https://arxiv.org/abs/1610.05492.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Ken MacLean. Voxforge. *Ken MacLean.[Online]. Available: http://www.voxforge.org/home.[Acedido em 2012]*, 2018.

Amirhossein Malekijoo, Mohammad Javad Fadaeieslam, Hanieh Malekijou, Morteza Homayounfar, Farshid Alizadeh-Shabdiz, and Reza Rawassizadeh. Fedzip: A compression framework for communication-efficient federated learning. *arXiv preprint arXiv:2102.01593*, 2021.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Aarti Singh and Jerry Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR, 20–22 Apr 2017. URL https://proceedings.mlr.press/v54/mcmahan17a.html.

Yue Niu, Saurav Prakash, Souvik Kundu, Sunwoo Lee, and Salman Avestimehr. Federated learning of large models at the edge via principal sub-model training, 2023.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: An asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210, 2015. doi: 10.1109/ICASSP.2015.7178964.

Tahseen Rabbani, Brandon Feng, Marco Bornstein, Kyle Rui Sang, Yifan Yang, Arjun Rajkumar, Amitabh Varshney, and Furong Huang. Comfetch: Federated learning of large networks on constrained clients via sketching, 2023.

Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *2007 15th European Signal Processing Conference*, pages 606–610, 2007.

Sebastian U. Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified sgd with memory. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 4452–4463, Red Hook, NY, USA, 2018. Curran Associates Inc.

Dimitris Stripelis, Umang Gupta, Greg Ver Steeg, and Jose Luis Ambite. Federated progressive sparsification (purge, merge, tune)+. *arXiv preprint arXiv:2204.12430*, 2022.

Vasileios Tsouvalas, Aaqib Saeed, Tanir Ozcelebi, and Nirvana Meratnia. Federated learning with noisy labels. *arXiv preprint arXiv:2208.09378*, 2022. doi: 10.48550/ARXIV.2208.09378.

P. Warden. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. *ArXiv e-prints*, apr 2018. URL https://arxiv.org/abs/1804.03209.

Chuhan Wu, Fangzhao Wu, Lingjuan Lyu, Yongfeng Huang, and Xing Xie. Communication-efficient federated learning via knowledge distillation. *Nature Communications*, 13(1), April 2022. ISSN 2041-1723. doi: 10.1038/s41467-022-29763-x. URL http://dx.doi.org/10.1038/s41467-022-29763-x.

Jiancheng Yang, Rui Shi, Donglai Wei, Zequan Liu, Lin Zhao, Bilian Ke, Hanspeter Pfister, and Bingbing Ni. Medmnist v2: A large-scale lightweight benchmark for 2d and 3d biomedical image classification. *arXiv preprint arXiv:2110.14795*, 2021.

A FedCompress **Algorithm**

We provide the pseudocode for `FedCompress` in Algorithm 1.

Algorithm 1 `FedCompress`: Federated Learning with Adaptive Weight Clustering and Server-Side Distillation for Model Compression. We develop a two-stage model compression approach to reduce the communication burden in FL and offer inference speedups in edge devices. FedAvg McMahan et al. (2017) is the base algorithm, whereas η_s and η_c refers to the learning rates of server and clients, respectively.

1: Server initialization of model with model weights θ^0, $C=C_{min}$
2: **for** $i = 1, \ldots, R$ **do**
3:      Randomly select K clients to participate in round i
4:      **for** each client $k \in K$ **in parallel do**
5:          $(\theta_k^{i+1}, \mathcal{E}_k^{i+1}) \leftarrow$ **ClientUpdate**(θ^i, C_i)
6:      **end for**
7:      $\theta^{i+1} \leftarrow \sum_{k=1}^{K} \frac{N_k}{N} \theta_k^{i+1}$, $\mathcal{E}^{i+1} \leftarrow \sum_{k=1}^{K} \frac{N_k}{N} \mathcal{E}_k^{i+1}$
8:      $\theta^{i+1} \leftarrow$ **SelfCompress**(θ^i, C_i)
9:      $C_{i+1} \leftarrow C_i + \mathrm{sgn} \left| \mathrm{MA}(\mathcal{E}^{i+1}) - \min_{j=1}^{P} \mathrm{MA}(\mathcal{E}^{i-j+1}) \right|$
10: **end for**
11: **procedure** CLIENTUPDATE(θ, C)
12:      **for** epoch $e = 1, 2, \ldots, E_c$ **do**
13:          **for** batch $b \in \mathcal{D}_l$ **do**
14:              $\theta \leftarrow \theta - \eta_c \cdot \nabla_\theta \left(\mathcal{L}_{ce}\left(p_\theta\left(b\right)\right) + \beta \cdot \mathcal{L}_{wc}\left(\theta, \mu, C\right) \right)$
15:          **end for**
16:      **end for**
17:      $\mathcal{E} \leftarrow \mathcal{E}(p_\theta(\mathcal{D}_u))$
18:      **return** $(\theta, \mathcal{E})$
19: **end procedure**
20: **procedure** SELFCOMPRESS(θ, C)
21:      **for** epoch $e = 1, 2, \ldots, E_s$ **do**
22:          $\theta^\star \leftarrow \theta$
23:          **for** batch $b \in \mathcal{S}$ **do**
24:              $\theta \leftarrow \theta - \eta_s \cdot \nabla_\theta \left(\mathcal{L}_{kl}\left(p_{\theta^\star}(b) \,||\, p_\theta(b)\right) + \beta_s \cdot \mathcal{L}_{wc}(\theta, \mu, C) \right)$
25:          **end for**
26:      **end for**
27:      **return** θ
28: **end procedure**
