Adaptive Pruning & Structural Aggregation for Network Bandwidth Optimization in Federated Learning

Vidushi Vashishth
Georgia Institute of Technology
vvashishth3@gatech.edu

Srihas Yarlagadda
Georgia Institute of Technology
syarlagadda37@gatech.edu

Aniruddha Mysore
Georgia Institute of Technology
animysore@gatech.edu

Aaditya Singh
Georgia Institute of Technology
asingh@gatech.edu

Abstract

The heterogeneity of edge devices and their limited and variable network and bandwidth resource availability make the adoption of federated learning (FL) for real-world applications a challenging problem. The benefit of user data privacy achieved through FL will not be realized if its adoption is not cost-effective and practical. To overcome these challenges, we propose Fed-SA: a novel FL methodology that performs network adaptive and distributed pruning of the model to better utilize the variable per-client network conditions while maintaining comparable accuracy to the baseline model. Furthermore, we propose a novel structural aggregation strategy on the server to aggregate weights from variably pruned models on the clients. Fed-SA conceptually generalizes to any structural pruning algorithm and CNN architecture in an FL setting. Through experimental evaluation, we demonstrate the efficacy of Fed-SA in navigating the trade-off between accuracy and network bandwidth use; this is supplemented with a qualitative comparison against state-of-the-art approaches such as PruneFL and Federated Pruning.

1 Introduction

The prevalent use of smart devices has enabled the increased adoption of neural network-based applications like speech recognition, machine translation, and image recognition. Innovation in edge computing has recently seen a rise in devices equipped with better hardware like mobile GPUs, higher RAM on mobile devices, and more. FL [12] enables distributed learning of neural networks on a network of edge devices instead of training on server-side data centers. This alleviates the privacy concerns of end-users as their data never leaves the edge devices. However, FL suffers from several limitations. It is constrained due to the limited compute and communication bandwidth available to these devices. On the client side, using resources efficiently is important as edge devices cannot allocate their entire compute and network to learning. On the server side, resource utilization and efficiency are key concerns — improving them can increase the number of clients served with the same hardware. Reducing the communication overhead of FL can also lead to lower data ingress and egress costs over the lifetime of the service, minimize energy use and emissions and benefit the environment. In this paper, we propose Fed-SA, a novel federated learning approach targeted at achieving the benefits of efficient network use.

Model architecture sizes have exponentially increased recently to extract better performance, often leading to over-parameterization. Transformer models like GPT-J for example, have as many as 6

Project repository: https://github.gatech.edu/cs8803smr-f22/repo-team14/

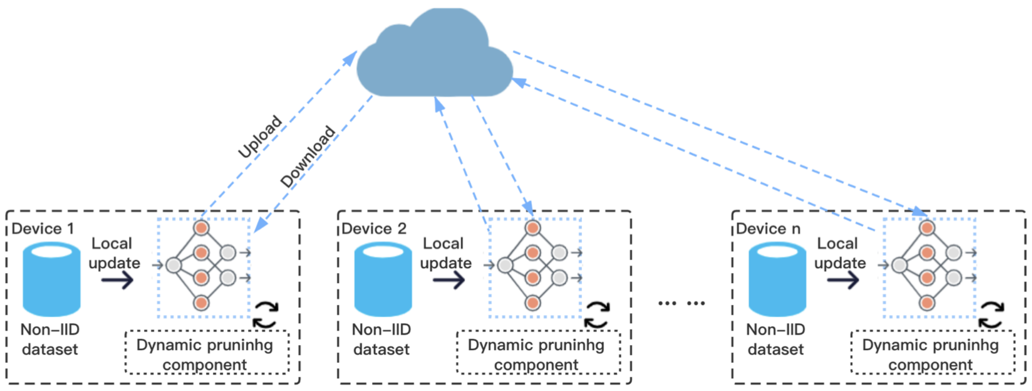

Figure 1: Structured Aggregation

billion parameters. Such over-parameterized networks are abundant with redundancies that can be exploited to improve model efficiency. Structural pruning is one approach that can help identify redundant parameters and reduce model size. Recent works [11] have explored centralized pruning to achieve model efficiency. In contrast, Fed-SA explores decentralized pruning in a network adaptive manner to take into account the network conditions per edge device. Through experimental evaluation, we demonstrate that Fed-SA can achieve better network bandwidth utilization in comparison to centralized pruning.

Fed-SA implements a novel model aggregation strategy. The server executing Fed-SA in an FL setting, cumulatively decides which nodes of the network to prune depending on the number of clients that vote the given node as less relevant to model performance. In this manner, Fed-SA can control the degree of the pruning by varying this threshold of minimum clients required to prune a node. Moreover, structurally pruning a model helps save network bandwidth as it directly reduces the parameter set's size. Non-structural pruning will not be as efficient in saving network bandwidth costs as they have to deal with sparse parameter matrices, which lack standardized storage and computation abstractions [11].

Our contributions can be summarized as follows:

- We propose a federated learning implementation of a structural pruning method for convolutional neural networks which can support different pruning factors for each client based on their network constraints.
- We devise a novel aggregation method to combine weights from each client as an extension to federated averaging, accounting for differences in weight matrix dimensions due to varying pruning factors.

2 Related Work

Neural Network Pruning.Neural network pruning has been a well-known technique to remove redundant parameters of a DNN for model compression, The earliest work can be traced back to 1980s ([20], [13], [9]). Most work in pruning focuses on the trade-off between accuracy and model sparsity in the inference stage. The idea is that smaller models not only have smaller SLOs but save on memory and computation bandwidth. A typical pruning process first calculates the importance scores of all parameters in a well-trained DNN and then removes parameters with lower scores. The importance scores can be derived based on the weight magnitudes ([9], [6]), the first-order Taylor expansion of the loss function ([20], [19]), the second-order Taylor expansion of the loss function ([13], [7]; [19]), and other variants ([16], [8]). Recent work by ([5]) suggested that a network consists of an optimal substructure of the original network, known as the "lottery ticket hypothesis". It showed that directly training the pruned network can reach a similar accuracy as pruning a pre-trained original network. Pruning can be further sub-divided into two categories, One is pruning at initialization, i.e., pruning the original full-size model before training. The pruning policy can be determined by evaluating the connection sensitivity ([14]), Hessian-gradient product ([22]), and synaptic flow ([21]) of the original model. Since such pruning does not involve the training data, the pruned model is not specialized for the training task, resulting in biased performance. The other category is dynamic sparse training ([18], [3], [4]). The pruned model structure is iteratively adjusted throughout the training process while maintaining the pruned model size at the desired sparsity. However, the pruning process is to adjust the model structure in a large search space, requiring memory-intensive operations, which is infeasible for resource-constrained devices.

Federated Neural Network Pruning. Federated learning has attracted great attention by enabling collaborative training across distributed and confidential datasets ([15]). Classical federated learning, represented by FedAvg ([17]), collects the locally updated on-device models rather than the raw data at the server for private knowledge sharing. Since data is locally stored and cannot be shared, the aforementioned pruning approaches that rely on training data can not be used in federated learning. Enlighten by pruning at initialization, ([23]) prunes the original full-size model at the server, and fine-tunes devices with their local data. Existing pruning at initialization approaches, such as SNIP ([14]), GraSP ([22]), and Syn Flow ([21]), can be directly converted to server-side pruning. However, server-side pruning usually results in significantly biased pruned models, especially for heterogeneous (non-iid) local data distributions. In another line of work Prune-FL ([10]) has shown that Adaptive pruning on edge can further help reduce the network bandwidth due to personalized model compression. Data-based network pruning not only leads to network bandwidth savings but helps maintain a balance between model performance and inference time, and accuracy. However, the above research suffers from large network costs due to the communication inefficiency of pruned models. Although PruneFL ([10]) reduces the local computational cost by finer pruning a coarse-pruned model rather than a full-size model, it still requires a large local network footprint to communicate the updated important scores of all parameters in the pruned model, PruneFL also does not consider the overall network budget in sampling/tuning the edge pruned networks. Therefore, existing federated neural network pruning fails to obtain a specialized edge model without bias and network-budget concerns, and we develop Fed-SA to achieve this.

3 Fed-SA Algorithm

The process is initialized with a pre-trained model that the server distributes to each of the clients. Each client trains this model for a set number of training epochs and then prunes the model locally. The degree of pruning can be varied to account for limited network capabilities. After pruning, the model weights and the indices of the dropped channels are transmitted back to the server.

The server uses the pruned indices from the clients to determine which channels to drop from the central model. This is decided by an aggregation fraction — the value describes the minimum number of cumulative training examples across clients that dropped a particular channel that is required to drop it in the central model. A FEDAVG based approach is used to then update the weights of the central model using the updated weights sent by the clients.

The updated central model is then distributed among the clients in the next round, and the process repeats. Through this procedure, we decrease the communication overhead both while sending the model from the server to the clients and while returning the updated weights.

Algorithm 1 FEDSA-CLIENT(j)

```
Require: $\eta_j^{(k)}$ for $k \in \mathbb{Z}_+$
1: Receive $\mathbf{W}_{P,j}^{(0)}$ from server
2: Send $n_j$ to server
3: for $k = 1, 2, \ldots$ do
4:     Receive $\mathbf{W}_{P}^{(k-1)}$ from server
5:     $\left(\mathbf{W}_{P,j}^{(k)}\right) \leftarrow \mathrm{SGD}_j\left(\mathbf{W}_{P,j}^{(k-1)}\right)$
6:     $\left(\mathbf{W}_{P,j}^{(k)}, \mathcal{M}_j^{(k)}\right) \leftarrow \mathrm{PRUNE}_j\left(\mathbf{W}_{P,j}^{(k-1)}\right)$
7:     Send $\mathbf{W}_{P,j}^{(k)}, \mathcal{M}_j^{(k)}, n_j$ to server
```

Given the nature of statistical heterogeneity in network-aware federated learning (FedSA), we make the following assumptions that can be readily relaxed in implementation: (a) the batch size b and the #epochs e are invariant across clients and across global aggregations, and (b) each client uses SGD for updates $\left(\mathbf{W}_{P_j}\right)$ between global aggregations.

Our proposed federated training algorithm is described via Algorithms 1 and 2. The steps for the server component of FEDSA are detailed under Algorithm 2 and the steps for a given client j are described under Algorithm 1.

Algorithm 2 FEDSA-SERVER

```
 1: Initialize $\mathbf{W}_P^{(0)} \longleftarrow \mathbf{W}_{Pretrained}$
 2: Receive $n_j$ from each client $j \in \{1, \ldots, N\}$
 3: Send $\mathbf{W}_P^{(0)}$ to each client
 4: for $k = 1, 2, \ldots$ do
 5:     Sample clients based on network heuristics in $\mathcal{H}$
 6:     Choose clients $\mathcal{C}_k \subseteq H$
 7:     Receive $\mathbf{W}_{P,j}^{(k)}$, $\mathcal{M}_j^{(k)}$ from each chosen client $j \in \mathcal{C}_k$
 8:     Compute $\gamma_j = n_j / \sum_{j \in \mathcal{C}_k} n_j$
 9:     for $l = 1, 2, ... \, Layers$ do
10:         for $c = 1, 2, ... \, Channels$ do
11:             Count = 0
12:             if $c_j$ dropped then
13:                 Count = Count + 1
14:             if Count > $Threshold$ then
15:                 $\mathcal{M}_{L_c}^{(k)} \leftarrow Drop \, \mathcal{M}_{j,L_c}^{(k)}$
16:     Aggregate $\mathbf{W}_P^{(k)} \leftarrow \sum_{j \in \mathcal{C}_k} \gamma_j \text{PRUNE}\left(\mathcal{M}^{(k)}, \mathbf{W}_{P,j}^{(k)}\right)$
17:     Send $\mathbf{W}_P^{(k)}$ to each client
```

Let n_i denote the number of input channels for the i^{th} convolutional layer and h_i/w_i be the height/width of the input feature maps. The convolutional layer transforms the input feature maps x_i into the output feature maps $x_i \in \mathcal{R}^{n_{i+1} \cdot n_{i+1} \cdot n_{i+1}}$, which are used as input feature maps for the next convolutional layer. This is achieved by applying n_{i+1} 3D filters $F_{i,j} \in \mathcal{R}^{n_i \cdot k \cdot k}$ on the n_i input channels, in which one filter generates one feature map. Each filter is composed by n_i 2D kernels $x_i \in \mathcal{R}^{k \cdot k}$ (e.g., 3 × 3). Kernel matrix together makes for all the filters $F_i \in \mathcal{R}^{n_i \cdot n_{i+1} \cdot k \cdot k}$. The number of operations of the convolutional layer is $n_{i+1} \cdot n_i \cdot k^2 \cdot h_{i+1} \cdot w_{i+1}$. As shown in Figure 1, when a filter $\mathcal{F}_{\rangle,|}$ is pruned, its corresponding feature map $x_{i+1,j}$ is removed, which reduces $n_i \cdot k^2 \cdot h_{i+1} \cdot w_{i+1}$ operations. The kernels that apply on the removed feature maps from the filters of the next convolutional layer are also removed, which saves an additional $n_{i+2} \cdot k^2 \cdot h_{i+2} \cdot w_{i+2}$ operations. Pruning m filters of layer i will reduce m/n_{i+1} of the computation cost for both layers i and $i + 1$.

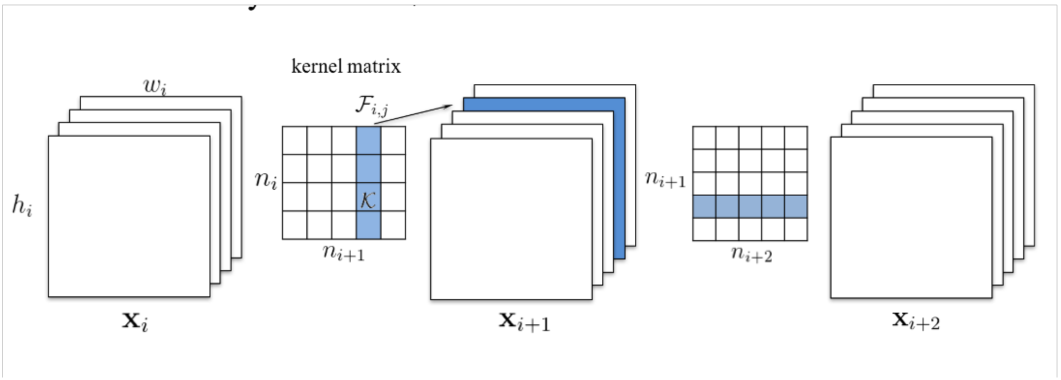

Figure 2: Adaptive pruning

4 Evaluation

4.1 Experimental Setup

Model, Dataset and Pruning Algorithm The ResNet-18 model was chosen for evaluation due to its suitability for the mobile device target deployment and the popularity of image tasks in machine learning on edge devices. The dataset used was CIFAR-10, with 60000 total colored images of 32x32 resolution. Random horizontal flips were used as a data augmentation method for training data, which was split between the clients. The Torch-Pruning open-source implementation of the channel-based pruning algorithm for CNNs was used for pruning the model at the client and the server, with modification to extract the channel indices for each round of pruning.

FL Implementation The Flower API [1] was used to implement a synchronous federated learning architecture, consisting of a central server communicating with multiple clients. Custom serialization/deserialization was required to transfer the server model to the clients and retrieve the client weights, along with the indices of the channels pruned by each client. A custom strategy extending the FedAvg base was implemented to allow for structural aggregation and server-side pruning.

Testbench The experiments were run on an Azure cluster with 2 virtual machines. The VM running the Flower server (and a portion of the clients) was equipped with 2 Standard D2s vCPUs and 8 GB of memory, while the VM running the rest of the clients was equipped with 4 Standard D4s vCPUs and 16 GB of memory.

Metrics Accuracy and network bandwidth usage were the metrics chosen to evaluate performance trade-offs. Network usage data was collected using a packet sniffer on the server port measuring total traffic. Accuracy was evaluated at each client over the full CIFAR-10 test set with 10,000 samples, weighted and aggregated using the number of training samples per client. This is reflective of a 'global accuracy' that is computed in a distributed fashion.

4.2 Pruning Algorithm Performance

We first evaluate the performance of the chosen pruning algorithm for one isolated instance. The pruning algorithm was run for multiple rounds starting with a pre-trained ResNet18 model, with one training epoch between each round of pruning. The results are summarized in Table 1. The aggregation fraction value used was 1.

Round #	Claimed Acc.	Observed Acc.	Model Size (# params)	Model Filesize (MB)
0 (Base)	0.9248	0.9241	11.1M	44.8
1	0.9229	0.8543	4.5M	18
2	0.9207	0.8053	1.9M	7.7
3	0.9176	0.7325	0.8M	3.4
4	0.9102	0.7397	0.4M	1.7
5	0.9011	0.7466	0.2M	0.8

Table 1: Model performance with pruning rounds.
*parameters in millions (M)

While reasonably high accuracy was achieved by each of the pruned models, the values were lower than the claimed performance. We observe that there is a significant decrease in model size with each iteration, and accuracy generally decreases. Below sizes of around 0.8 MB the performance diminishes severely.

4.3 Adaptive Pruning Performance

The network adaptive Fed-SA algorithm was tested to determine its position w.r.t the accuracy and network bandwidth trade-off. 8 clients were divided into three groups, with different degrees of pruning to simulate different network constraints. The constants used for the testing configuration are summarized in table 4. The model used for initialization was the round 3 centrally pruned model as described in the previous section. The same values are used in all following experiments unless explicitly specified.

Model Size (pruning rounds)	Clients	Network Bandwidth (MB)	Time (s)	Per-VM Client Split	Accuracy		
11.1 M (0)	2	23.12	4269.7	2	0.78	0.87	0.85
0.2 M (5)	2	2.93	445.80	2	0.71	0.76	0.78
0.2 M (5)	8	7.31	705.20	8	0.65	0.76	0.79
0.2 M (5)	24	10.11	598.85	8/16	0.73	0.77	0.80
0.2 M (5)	36	21.96	777.77	8/8/16	0.77	0.82	0.82

Table 2: Network bandwidth consumption during federated training of pruned models

Round #	Accuracy
1	66.4
2	70.73
3	70.585
4	71.792
5	72.49
Total Network use: 6.7MB	

Table 3: Adaptive Pruning performance

The accuracy of network adaptive pruning for each round shows convergence (table 3), and is comparable to that of the centrally pruned model. Further, we observe that the total network usage is better than the centrally pruned model, owing to the smaller size of client updates and the central model. The total network use also corresponds to a larger number of rounds in the adaptive pruning case, displaying the extent of savings. This validates our hypothesis.

Config. parameter	Value
Number of rounds	3
Batch size	64
Training epochs (per call)	3
Pruning factor (for each client group)	0.5, 0.75, 1
Learning rate step size	20
Number of clients	8 (2+3+2)

Table 4: Constants used for evaluation

4.4 Effect of Aggregation Fraction

The aggregation fraction is a float value between 0 and 1 that determines how aggressively the central model is pruned given a set of pruning indices obtained from the clients. A value of 1 indicates that a channel is to be pruned only if every client has pruned it (least aggressive), while a value of 0 indicates that a channel is pruned if any client has pruned it (most aggressive). The effect of varying the aggregation fraction is shown in figure 3.

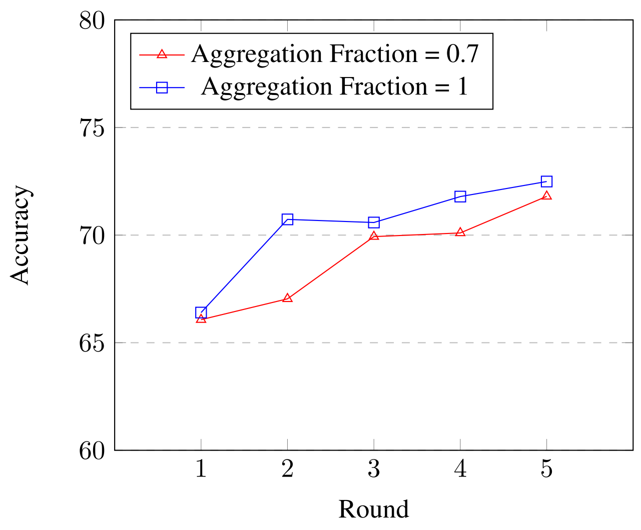

Figure 3: Accuracy variation over 5 rounds, with different values of the threshold aggregation fraction.

4.5 Early stopping

We finally test the effect of early stopping — stopping the central server pruning after a certain number of rounds to preserve accuracy, while continuing pruning at the clients to keep communication overheads minimized. For this test, pruning at the server was stopped after 3 rounds. The results are in table 5.

The uplift in accuracy with additional training epochs suggests that additional rounds of training can be used to improve accuracy without server side pruning, if desired. This results in greater flexibility.

Round #	Accuracy
1	68.43
2	70.59
3	69.75
4	71.33
5	72.96
6	74.385
7	73.24
8	74.41
9	72.76
10	73.81
Total Network use: 15.58MB	

Table 5: Early stopping performance

5 Conclusion

Through our evaluation, we have proven our hypothesis that network-adaptive pruning can be used to efficiently utilize network bandwidth while maintaining accuracy in a federated learning deployment. In addition to decreasing the communication overhead, Fed-SA can help mitigate stragglers by decreasing the computational load at each edge client as a result of the smaller model size. This leads to multiple benefits in energy efficiency, network use and computational efficiency across the board. Through further tuning of pruning factors and the aggregation threshold, accuracy can be better preserved while yielding the same performance.

6 Future Work

Aggregation for asynchronous federated learning Fed-SA relies on clients to prune the same server model in each iteration, and the client pruning indices are used in conjunction with this central model to determine which indices to drop. In order to allow for aggregation in an asynchronous federated learning setting, this mechanism needs to be changed to keep track of the history of central models and associate client responses with their corresponding rounds to derive context. The ability to modify the aggregation strategy independently allows this to be added while avoiding system-wide changes and complex integration.

Privacy preserving structural aggregation Methods such as Secure Aggregation [2] have been developed to address privacy concerns in federated averaging. Since client weight matrices are no longer of the same dimension, this protocol cannot be trivially applied to Fed-SA — specializations such as using the central model to determine weight masks need to be implemented at the cost of extra computation at the server.

Sampling strategies The extra headroom in terms of network use that is afforded by adaptive pruning opens up the possibility of sampling a larger number of clients, as well as different strategies for sampling from each client population based on their network constraints. This can be used to address problems like wasted computation due to dropouts and biases in model prediction.

References

[1] Daniel J. Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Titouan Parcollet, and Nicholas D. Lane. Flower: A friendly federated learning research framework. *CoRR*, abs/2007.14390, 2020.

[2] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 1175–1191, New York, NY, USA, 2017. Association for Computing Machinery.

[3] Tim Dettmers and Luke Zettlemoyer. Sparse networks from scratch: Faster training without losing performance. *ArXiv*, abs/1907.04840, 2019.

[4] Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. *ArXiv*, abs/1911.11134, 2019.

[5] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv: Learning*, 2018.

[6] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *arXiv: Computer Vision and Pattern Recognition*, 2015.

[7] Babak Hassibi and David G. Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *NIPS*, 1992.

[8] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *J. Mach. Learn. Res.*, 22:241:1–241:124, 2021.

[9] Janowsky. Pruning versus clipping in neural networks. *Physical review. A, General physics*, 39 12:6600–6603, 1989.

[10] Yuang Jiang, Shiqiang Wang, Bongjun Ko, Wei-Han Lee, and Leandros Tassiulas. Model pruning enables efficient federated learning on edge devices. *IEEE transactions on neural networks and learning systems*, PP, 2019.

[11] Yuang Jiang, Shiqiang Wang, Victor Valls, Bong Jun Ko, Wei-Han Lee, Kin K. Leung, and Leandros Tassiulas. Model pruning enables efficient federated learning on edge devices, 2019.

[12] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara

Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Mariana Raykova, Hang Qi, Daniel Ramage, Ramesh Raskar, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and open problems in federated learning, 2019.

[13] Yann LeCun, John S. Denker, and Sara A. Solla. Optimal brain damage. In *NIPS*, 1989.

[14] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Snip: Single-shot network pruning based on connection sensitivity. *ArXiv*, abs/1810.02340, 2018.

[15] Ang Li, Jingwei Sun, Binghui Wang, Lin Duan, Sicheng Li, Yiran Chen, and Hai Helen Li. Lotteryfl: Empower edge intelligence with personalized and communication-efficient federated learning. *2021 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 68–79, 2021.

[16] Christos Louizos, Max Welling, and Diederik P. Kingma. Learning sparse neural networks through l0 regularization. *ArXiv*, abs/1712.01312, 2017.

[17] H. B. McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *International Conference on Artificial Intelligence and Statistics*, 2016.

[18] Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H. Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature Communications*, 9, 2017.

[19] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11256–11264, 2019.

[20] Michael C. Mozer and Paul Smolensky. Skeletonization: A technique for trimming the fat from a network via relevance assessment. In *NIPS*, 1988.

[21] Hidenori Tanaka, Daniel Kunin, Daniel L. K. Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. *ArXiv*, abs/2006.05467, 2020.

[22] Chaoqi Wang, ChaoQi Wang, Guodong Zhang, and Roger Baker Grosse. Picking winning tickets before training by preserving gradient flow. *ArXiv*, abs/2002.07376, 2020.

[23] Wenyuan Xu, Weiwei Fang, Yi Ding, Meixia Zou, and Naixue N. Xiong. Accelerating federated learning for iot in big data analytics with pruning, quantization and selective updating. *IEEE Access*, 9:38457–38466, 2021.