Optimizing Model Pruning in Decentralized Learning Networks with DFL-Trim

Optimizing Model Pruning in Decentralized Learning Networks with DFL-Trim

Andrea Pinto * Alessandro Masci * † Alessio Sacco † Guido Marchetto † Flavio Esposito *
* *Department of Computer Science*, Saint Louis University, USA
† *Department of Control and Computer Engineering*, Politecnico di Torino, Italy

***Abstract*—In recent decades, applications in environmental sustainability, education, and housekeeping have become increasingly distributed and sophisticated, leveraging a wide range of devices to perform complex tasks. While a large number of agents can reduce computation time, managing these distributed systems presents significant challenges due to resource constraints such as power consumption and storage. To address this, the literature has explored various model compression techniques, such as pruning, to optimize performance in distributed environments. In this paper, we propose DFL-Trim, a solution for trimming models in Decentralized Federated Learning (FL) that meets network constraints while maintaining satisfactory performance. We demonstrate how pruning can be implemented in decentralized settings, analyze its effect on bandwidth usage, and discuss the trade-offs between compression and model accuracy.**

I. Introduction

The Internet of Things (IoT) paradigm is driving the development of modern companies and smart cities, enabling connections between distributed devices, such as mobile phones, tablets, and computers, which sense and transfer data from external environments to serve end users. To enable distributed learning across IoT devices, Federated Learning (FL) [1] offers a solution for training machine learning models like Deep Neural Networks (DNN) while preserving data privacy by enabling global model training without sharing private data [2]. Two main training paradigms have proven efficient in this context: centralized and decentralized. While centralized training is the traditional approach for FL, decentralized training is gaining momentum and has shown comparable effectiveness to centralized methods [1], [3]–[5]. Decentralized training communication patterns, on the other hand, increase bandwidth demands, especially as the density of node connections in the network topology grows [6], but make the learning process fit modern IoT applications, where a central server (a Parameter Server orchestrator) may not always be a feasible solution.

Traditional FL settings are unsustainable due to heterogeneous resource constraints and bandwidth variability. This is particularly relevant given the wide applicability and necessity of ML techniques across domains such as Industry 5.0, healthcare, and vehicular IoT, where devices are often limited in power, computing, and storage [1], [7], [8]. To address the constrained nature of devices participating in FL and to efficiently utilize limited bandwidth, *Model Compression Technique* (MCT) [9] can be employed to maintain acceptable accuracy levels while reducing resource usage and bandwidth consumption [10]. Pruning is one of the most widely adopted MCT [9], recognized as an effective method for reducing memory usage and minimizing communication overhead, particularly in terms of bandwidth requirements during distributed training. This approach achieves these benefits while maintaining performance comparable to the original DNN and reduces inference times. Different systems have validated the effectiveness of pruning techniques in traditional centralized federated learning settings [6], [11], [12]. However, the decentralized and resource-constrained nature of IoT networks introduces additional challenges that remain to be addressed.

To this end, we propose DFL-Trim, a novel approach for trimming (pruning) decentralized federated learning models—specifically engineered for resource-constrained IoT environments. Unlike conventional pruning techniques that target centralized settings, DFL-Trim introduces two key innovations: (i) Adaptive Pruning for Decentralized Scenarios: It tailors pruning strategies to account for limited bandwidth, power constraints, and heterogeneous data typical of IoT devices. (ii) Versatile Integration with Network Architectures: It seamlessly supports both consensus-based and ring topologies, ensuring efficient convergence and minimized communication overhead. We benchmark our solution against the CIFAR-10 (to isolate the effects of various pruning methods) dataset to standardize measurements of accuracy, inference time, and data exchange. In our experiments, we compare DFL-Trim across two network configurations—a ring topology with two peer connections per client and a consensus topology with 30% connectivity. Our results show that while both configurations achieve similar accuracy under a specific pruning regime, they differ significantly in terms of message exchange and convergence time, with the choice of pruning method critically affecting final performance.

II. Related Work

Training in FL can follow centralized or decentralized approaches, but pruning, despite its success in centralized settings, remains underexplored in decentralized scenarios.

Centralized vs. Decentralized. The client topology significantly impacts FL performance. While centralized training is common, decentralized methods can match its performance and may be necessary in certain scenarios [13]. Consensus-based (CB) methods have emerged as alternatives to parameter server

and ring-all-reduce approaches for large-scale training [3]. Interestingly, sparser topologies may converge faster, not due to reduced communication, but by alleviating the straggler effect. This paper examines both ring-based and CB training under varying connection densities.

Model Pruning. Model Compression Techniques (MCTs), especially pruning and quantization [14], [15], improve FL by reducing model size and communication overhead. Pruning has proven effective for memory and inference efficiency [16], [17], and locally applied pruning can also aid convergence. However, in FL, where clients have private data, pruning may hinder convergence unless adapted. Recent algorithms address this, showing that pruning can reduce communication while preserving performance [6], [11], [12]. While these approaches are effective, they are highly specialized for specific scenarios and fail to converge in diverse conditions. In this paper, we propose an aggregation schema that can fit diverse underlying topologies and study how various pruning techniques can be applied alongside our proposed solution.

III. Modeling Distributed Training and Pruning

Pruning is a key technique used to optimize neural networks by reducing model size and computational complexity. However, in distributed training settings like in FL, it can reduce the bandwidth utilization given the reduced number of weights to exchange. We start introducing different pruning variants and then describe our DFL-Trim training.

A. Unstructured and Structured Pruning

Definition 1 (Unstructured Pruning): Given neural network weights $W = \{w_0, w_1, \ldots, w_K\}$, a dataset $D = \{(x_i, y_i)\}_{i=1}^{N}$ consisting of input-output pairs, and a desired number of non-zero weights k (where $k < K$), unstructured pruning can be formulated as the following constrained optimization problem:

$$\min_{W} L(W; D) = \frac{1}{N} \sum_{i=1}^{N} l(W; (x_i, y_i))$$
$$\text{subject to} \quad \|W\|_0 \leq k$$

In practice, especially for small to medium-sized models, unstructured pruning typically does not set the weights directly to zero. Instead, it applies a binary mask M to the weights, where M indicates which weights are retained. This approach modifies the optimization problem to:

$$\min_{W,M} L(W \odot M; D) = \frac{1}{N} \sum_{i=1}^{N} l(W \odot M; (x_i, y_i))$$
$$\text{subject to} \quad \|M\|_0 \leq k$$

Here, $W \odot M$ denotes element-wise multiplication of the weights and the mask. Typically, the network is retrained with the masks fixed, ensuring that the pruned weights do not participate in further training. Unstructured pruning allows for the removal of weights from any location within the network, resulting in irregular sparsity patterns.

Definition 2 (Structured Pruning): Given a neural network with structural elements $S = \{s_1, s_2, \ldots, s_L\}$, where each s_i represents a set of channels, filters, neurons, or transformer attention heads in layer i, and a desired pruning ratio r, structured pruning seeks to identify a subset $S' = \{s'_1, s'_2, \ldots, s'_L\}$ where $s'_i \subseteq s_i$ for each $i \in \{1, \ldots, L\}$. The objective is to minimize performance degradation while maximizing computational speedup under the given pruning ratio.

Mathematically, structured pruning can be formulated as the following optimization problem:

$$\min_{S'} L(S'; D)$$
$$\text{subject to} \quad \frac{|S'|}{|S|} \geq 1 - r$$

where: $L(S'; D)$ represents the loss function of the pruned network S' on dataset D. $|S'|$ and $|S|$ denote the number of structural elements after and before pruning, respectively. r is the desired pruning ratio.

In practice, structured pruning removes entire filters, channels, transformer attention heads, or even whole layers.

B. Global and Local Pruning

Pruning strategies can further be categorized based on the scope of pruning operations: global pruning and local pruning. These strategies can be applied to both unstructured and structured pruning, influencing how and where parameters or structures are removed within the network.

The difference between global [18] and local [19] pruning lies in whether structures are removed from a subset or all available structures of a network. A major limitation of local pruning is that setting a pre-defined prune ratio for each layer can be complex and lead to sub-optimal sparsity. To simplify, local pruning often uses a consistent prune ratio across layers. In contrast, global pruning automatically generates a varying prune ratio for each layer. Global and local pruning strategies influence both unstructured and structured pruning approaches. In the context of unstructured pruning, local pruning applies a fixed prune ratio to each layer, removing individual weights based on their importance within each specific layer. This method maintains uniform sparsity across layers but may not account for the varying significance of different layers. On the other hand, global unstructured pruning considers all weights across the entire network and removes the least important weights irrespective of their layer, allowing for a more flexible and potentially more effective distribution of sparsity.

Global pruning tends to achieve better overall sparsity by allocating more pruning to less critical layers, whereas local pruning may struggle to balance pruning ratios across layers, potentially leading to either over-pruning or under-pruning in certain parts of the network. We formalize the different

techniques as: Global Unstructured Pruning (GUP), Local Unstructured Pruning (LUP), Local Structured Pruning (LSP). (Note that these are available in the Pytorch library, and Global Structured pruning is not implemented as it is too resources expensive.) Specifically, we will evaluate the effects of these techniques varying the topology sparsity interconnecting the workers joining the training process to assess how these techniques are comparable over different training scenarios.

C. Training Pipeline

Pruning after training is the most widely adopted pruning strategy, based on the prevailing belief that pretraining a dense network is essential for identifying an effective subnetwork [20]. This pipeline encompasses three main stages: pretraining, pruning, and retraining.

Algorithm 1 outlines the details of DFL-Trim, which begins after the pretraining step (Line 3). During pretraining, a densely initialized network $f(x; W_0)$ is trained until it converges to $f(x; W_t)$. We define a topology T, represented as a two-dimensional adjacency matrix, which indicates whether node i is logically connected to node j. We assume that the topology contains no islands, ensuring that each client has at least one connection. Furthermore, we assume that all clients in the pool N have access to the same model.

The consensus algorithm employed follows the best practices of consensus-based algorithms [21]. The algorithm runs for a specified number of pruning iterations I (Lines 4-20) and communication rounds R (Lines 5-19) to reach a target accuracy, with Algorithm 1 being executed at each iteration. Each client exchanges its locally pruned model with its direct neighbors and applies local pruning using one of the previously mentioned techniques (GUP, LUP, LSP) with a specified pruning rate (Lines 7-11). The selected pruning technique removes specific weights (or alternatively, filters, neurons, etc.) with minimal impact on performance, resulting in a pruned network denoted by $f(x; W'_t \odot M')$, where W'_t represents the pruned weights and M' the corresponding masks. Pruning can be performed either once (one-shot pruning) or iteratively multiple times. After pruning the local network, each client sends the results to its neighbors (Line 11). The neighbors then average the received weights, apply pruning again, and send back the averaged and updated model to their neighbors (Lines 14-17). This operation is repeated for a predefined number of iterations I (Line 4), ultimately leading to a globally consistent model. Upon completion of the final training phase, the network's accuracy and inference time are recorded for evaluation purposes.

Algorithm 1 DFL-Trim: Decentralized FL Pruning

1: **Input:** Number of clients N; Topology T; Initial model W_0; Pruning threshold θ; Communication rounds R; Local epochs E; Pruning rate ρ; Pruning iterations I;
2: **Output:** Pruned global model W'
3: Initialize $W_k \leftarrow W_0$ and $M_k \leftarrow \mathbf{1}$ for all $k \in \{1, \ldots, N\}$
4: **for** each iteration $i = 1$ to I **do**
5: **for** each communication round $r = 1$ to R **do**
6: **for all** clients k in parallel **do**
7: Train W_k for E epochs on local data
8: $W_k \leftarrow W_k \odot M_k$
9: $M_k \leftarrow \text{Prune}(W_k, \theta)$
10: $W'_k \leftarrow W_k \odot M_k$
11: Send W'_k to neighbors based on T
12: **end for**
13: **for all** clients k in parallel **do**
14: Receive $\{W'_j\}$ from neighbors j in T_{kj}
15: $W_k \leftarrow \frac{W_k + \sum_{j \in \mathcal{N}_k} W'_j}{|\mathcal{N}_k| + 1}$
16: $M_k \leftarrow \text{Prune}(W_k, \theta)$
17: $W_k \leftarrow W_k \odot M_k$
18: **end for**
19: **end for**
20: **end for**

IV. Results

In this section, we describe the details of the implementation testbed and present the results obtained from training with the DFL-Trim algorithm. The results include accuracy and inference time after the distributed training process, considering different pruning techniques (Section III) and varying the logical topology density. Specifically, we compare training outcomes in a consensus-based randomly generated topology with 30% connectivity against a ring topology with 12 clients. Additionally, we evaluate bandwidth utilization patterns resulting from the consensus algorithm. We exclude the dynamic churn behavior of nodes joining or leaving the training process, as well as temporary disconnections from the network—a common characteristic of systems where nodes are not consistently online or available, such as mobile networks, peer-to-peer systems, or edge computing environments. The convergence of a decentralized training system in this more complex scenario has already been demonstrated [21].

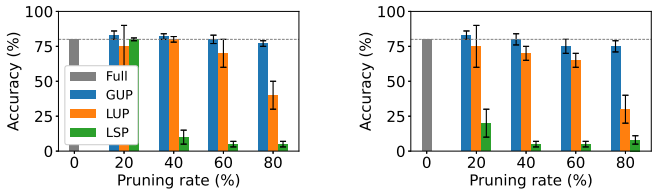
System Implementation: We simulate a training scenario using RTX 6000 GPUs available on the Chameleon Cloud [22], emulating workers with Docker containers. We perform training by varying the pruning techniques (Section III) and the percentage of weights pruned (20%, 40%, 60%, or 80%), focusing on ResNet18 and TinyYoloV2 models for an image classification task using the CIFAR-10 dataset. Specifically, we divide the dataset into two parts: the first part is used to train the model globally, and the second part is distributed as private data to each client, with each client applying a distinct image transformation to the entire dataset. This approach ensures that each client possesses sufficient input samples to achieve an acceptable accuracy baseline within a reasonable number of epochs, while concurrently training the local model on a unique dataset. Consequently, we consider a non-IID data distribution in the distributed training process. The pruning strategies implemented in PyTorch's prune library creates a

bitmask representing the weights to be considered during the forward step. Pruning approaches can be local, removing a fixed percentage of parameters per layer, or global, selecting a fraction across all layers. Weight pruning is also classified as structured if it preserves the network's shape for better hardware efficiency or unstructured if it removes parameters without a specific pattern.

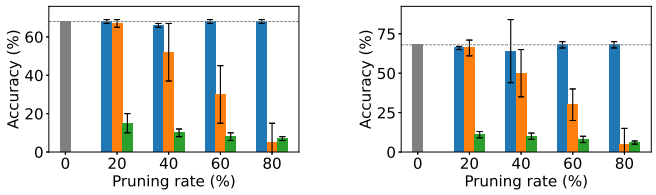
Accuracy drops with heavy pruning: In the first set of experiments, we measure the achieved accuracy after executing decentralized training following Algorithm 1. Specifically, Figure 1 illustrates the accuracy degradation across different pruning percentages (20% to 80%) and pruning techniques (GUP, LUP, LSP) within the two network topologies. Figures 1a and 1b display the accuracy results for the ResNet18 DNN, while Figures 1c and 1d present the results for the TinyYoloV2 DNN, both trained on the CIFAR-10 dataset. It is evident that the pruning rate affects accuracy in both cases. Specifically, an 80% pruning rate significantly reduces the DNN size, leading to a drastic drop in delivered accuracy. However, these experiments demonstrate that our decentralized training approach impacts accuracy differently depending on the pruning ratio. The impact is less pronounced with GUP pruning across all experiments, neural networks, and topology settings. GUP often outperforms LUP and LSP in maintaining model accuracy because it can selectively prune the least important weights across the entire network, allowing more critical layers to retain their capacity. LUP performs comparably but may not achieve the same level of compression without some loss in performance. LSP tends to be less efficient in this range since removing entire structures can disrupt the network's representational capacity more significantly. In conclusion, the choice of pruning method significantly affects the final accuracy but GUP is able to preserve the accuracy even for considerable model compression.

Inference Time Reduction: We then conducted this experiment to demonstrate how pruning impacts inference time. Figure 2 illustrates how the inference time of the trained models changes with different pruning ratios. The trends for both DNN architectures (ResNet18 2a and TinyYoloV2 2b) demonstrate a decrease in inference time, achieving up to a 20% reduction compared with the baseline full DNN without pruning at the highest pruning ratio. However, this reduction comes at the cost of decreased accuracy. Another factor that might impact inference time is the hardware used for testing. Specifically, we collected these results on the same GPU used for training; thus, our experiments are bounded by our experimental settings.

Model size impact: Pruning reduces a neural network's size by removing weights, but even with the same pruning rate, the final model size can vary depending on the approach used. Figure 3 shows the pruning impact for the two DNN analyzed. GUP removes the least important weights across the entire network, leading to an optimal reduction in overall parameters. However, since it prunes sparsely across all layers, some layers may retain a high number of parameters while others become

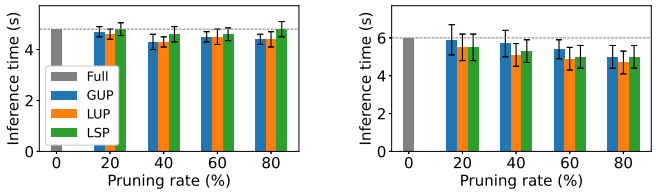


(a) ResNet18 Ring (b) ResNet18 Consensus



(c) TinyYoloV2 Ring (d) TinyYoloV2 Consensus

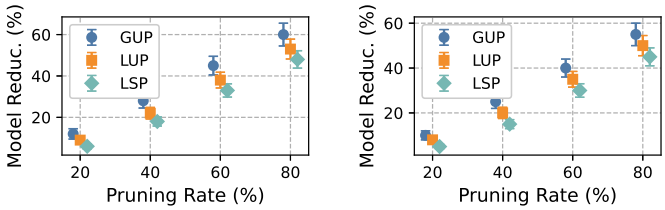
Fig. 1: Higher pruning rate negatively affects the performance of both models by reducing the number of trainable weights, which in turn limits generalization. However, GUP pruning maintains higher accuracy, even at a pruning rate of 80%. Topology density does not affect the accuracy.



(a) TinyYoloV2 (b) ResNet18

Fig. 2: Inference behavior varies across different DNN depending on the pruning ratio. Generally, a higher pruning ratio leads to reduced inference time, but this comes at the cost of decreased accuracy.

much smaller. This flexibility maximizes parameter reduction while preserving accuracy, but it results in irregular sparsity that is challenging for hardware to exploit. LUP, by contrast, applies pruning evenly across all layers. This ensures that no single layer is overly reduced, preserving a more uniform structure. However, because it enforces a per-layer pruning rate rather than targeting global redundancies, it may leave some unnecessary parameters intact, leading to a larger model compared to GUP at the same pruning rate. LSP removes entire structures, such as neurons, filters, or channels. While this creates highly efficient, hardware-friendly sparsity, the removal of entire structures means that some parts of the model may be pruned more aggressively than others. Since structured pruning eliminates full units rather than individual weights, the remaining model may retain more parameters than an equivalent unstructured pruning approach, resulting in different final sizes despite the same nominal pruning rate. Thus, the final model size differs because GUP optimally selects parameters globally, LUP enforces uniformity across layers, and LSP removes entire



(a) TinyYoloV2 (b) ResNet18

Fig. 3: Pruning does impact the model size, which is a desired behavior in distributed training given the constrained bandwidth and resources available. This would lead to reduced transmission times while sacrificing accuracy.

structures, each affecting redundancy and efficiency in distinct ways.

Bandwidth overhead of consensus-based topology. Given the distributed consensus approach evaluated in our study, we quantify the amount of data exchanged among clients during training, where each client averages the model with its neighbors. The data transfer complexity is given by: $O(kIM)$, where k is the number of neighbors per node, I is the number of consensus iterations, and M is the model size. The latency depends on the network diameter: $O(d)$, where d represents the longest shortest path in the communication graph. Thus, the main parameters impacting convergence are related to the size of the model, which can be mitigated using pruning ratios and techniques. Moreover, a reduced topology density will lead to lower bandwidth usage.

V. Conclusion

By proposing *DFL-Trim*, we show how pruning can be effectively utilized to train models in decentralized federated learning settings. While various pruning techniques exist, increasing the pruning rate can preserve model accuracy with *DFL-Trim*. We also analyze the trade-offs between accuracy, inference time, and model size following decentralized training, which we implement in consensus-based and ring-based topologies. Our results confirm the convergence of DFL-Trim algorithm across different pruning techniques. Among them, Global Structured Pruning emerges as the most effective approach across the scenarios analyzed. We observe that the density of the topology has no significant impact on training accuracy, while it does impact bandwidth utilization and communication overhead, which may indirectly affect learning efficiency.

Acknowledgment

This work has been supported by the National Science Foundation (NSF) Awards OAC # 2201536 and # 2430236 and by the European Union - Next Generation EU under the Italian National Recovery and Resilience Plan (NRRP), Mission 4, Component 2, Investment 1.3 (PE00000001 - program "RESTART"). The work of Alessandro Masci was conducted in the Department of Computer Science at Saint Louis University.

References

[1] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, and H. Vincent Poor, "Federated learning for internet of things: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, p. 1622–1658, 2021.

[2] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," 2023.

[3] G. Neglia, C. Xu, D. F. Towsley, and G. Calbi, "Decentralized gradient methods: does topology matter?," in *International Conference on Artificial Intelligence and Statistics*, 2020.

[4] Q. Luo, J. Lin, Y. Zhuo, and X. Qian, "Hop: Heterogeneity-aware decentralized training," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '19, p. 893–907, ACM, 2019.

[5] L. Pappone, A. Sacco, and F. Esposito, "On traffic matrix estimation via super-resolution and federated learning," *IEEE Transactions on Network and Service Management*, 2024.

[6] Z. Chen, D. Li, J. Zhu, and S. Zhang, "DACFL: Dynamic Average Consensus-Based Federated Learning in Decentralized Sensors Network," *Sensors*, vol. 22, p. 3317, Apr. 2022.

[7] O. Aouedi, T.-H. Vu, A. Sacco, D. C. Nguyen, K. Piamrat, G. Marchetto, and Q.-V. Pham, "A survey on intelligent internet of things: Applications, security, privacy, and future directions," *IEEE Communications Surveys & Tutorials*, pp. 1–1, 2024.

[8] O. Aouedi, A. Sacco, L. U. Khan, D. C. Nguyen, and M. Guizani, "Federated learning for human activity recognition: Overview, advances, and challenges," *IEEE Open Journal of the Communications Society*, vol. 5, pp. 7341–7367, 2024.

[9] H. Cheng, M. Zhang, and J. Q. Shi, "A survey on deep neural network pruning-taxonomy, comparison, analysis, and recommendations," 2024.

[10] Y. Jiang, S. Wang, V. Valls, B. J. Ko, W.-H. Lee, K. K. Leung, and L. Tassiulas, "Model pruning enables efficient federated learning on edge devices," 2022.

[11] S. Bibikar, H. Vikalo, Z. Wang, and X. Chen, "Federated Dynamic Sparse Training: Computing Less, Communicating Less, Yet Learning Better," 2021.

[12] Q. Long, C. Anagnostopoulos, S. P. Parambath, and D. Bi, "FedDIP: Federated Learning with Extreme Dynamic Pruning and Incremental Regularization," 2023.

[13] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, p. 5336–5346, Curran Associates Inc., 2017.

[14] W. Shao, M. Chen, Z. Zhang, P. Xu, L. Zhao, Z. Li, K. Zhang, P. Gao, Y. Qiao, and P. Luo, "Omniquant: Omnidirectionally calibrated quantization for large language models," 2024.

[15] J. Cheng, J. Wu, C. Leng, Y. Wang, and Q. Hu, "Quantized cnn: A unified approach to accelerate and compress convolutional networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 10, pp. 4730–4743, 2018.

[16] S. Hanson and L. Pratt, "Comparing biases for minimal network construction with back-propagation," in *Advances in Neural Information Processing Systems* (D. Touretzky, ed.), vol. 1, Morgan-Kaufmann, 1988.

[17] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," 2016.

[18] T.-W. Chin, R. Ding, C. Zhang, and D. Marculescu, "Towards efficient model compression via learned global ranking," 2020.

[19] J.-H. Luo, J. Wu, and W. Lin, "Thinet: A filter level pruning method for deep neural network compression," 2017.

[20] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the value of network pruning," 2019.

[21] I. Hegedüs, G. Danner, and M. Jelasity, "Gossip learning as a decentralized alternative to federated learning," in *IFIP International Conference on Distributed Applications and Interoperable Systems*, 2019.

[22] K. Keahey *et al.*, "Lessons learned from the chameleon testbed," in *USENIX Annual Technical Conference (ATC '20)*, pp. 219–233, USENIX Association, July 2020.