Optimized Federated Learning Through Watts-Strogatz Network Priors

Mihailo Ilić[1,*], Miloš Savić[1] and Mirjana Ivanović[1]

[1]*University of Novi Sad, Faculty of Sciences, Trg Dositeja Obradovića 3, Novi Sad 21000, Republic of Serbia*

Abstract

One of the main applications of federated learning (FL) is in massively distributed Internet of Things (IoT) environments. These environments often involve the use of devices with limited processing and storage capabilities. As learning in FL is most commonly done at the edge, this opens up various resource related challenges. Consequently, exploring optimization options which reduce the computational load on these devices is a crucial topic. One approach to achieving this is by reducing the number of learnable parameters in the machine learning (ML) model being trained. This paper explores the use of Watts-Strogatz networks as a baseline for hidden layer architectures of artificial neural networks (ANNs) as a strategy for network pruning prior to training. This approach reduces the number of computations needed during training, while also reducing the size of transmitted messages between the edge nodes and the FL server. By optimizing network architectures used by FL models, resource consumption may be significantly reduced, thus bringing this learning paradigm to more resource-constrained environments.

Keywords

Model Sparsification, Federated Learning, Network Priors, Optimization

1. Introduction

IoT environments collect vast amounts of data. As such, distributed learning tasks are well suited for them, which can be utilized to extract knowledge from these intensive data collection workloads. As large systems can have even hundreds of thousands of IoT edge devices, centralized data collection may not always be an ideal option, or even possible, which is why distributed learning is a necessary alternative. Federated learning is a distributed learning paradigm with a broad range of applications and use cases, one of which is in IoT. The most common way of implementing FL is by having data collection and learning done directly at the edge, while the learning process of the global machine learning model is coordinated by a central server [1, 2].

The number of participants may vary from a couple of thousand to hundreds of thousands of devices (edge nodes). These edge nodes may be heterogeneous in terms of processing power, data availability, data quality, etc. Consequently, numerous challenges are introduced like communication latency, the size of messages being communicated, optimal edge node sampling, communication bottlenecks at the server, and much more.

Since this learning paradigm is a great fit for ML in IoT, optimization of this process through various means like message quantization, optimized sampling, etc. is covered in related work. The work outlined in this paper, however, presents a different approach. Taking into account the computational cost of training deep models on IoT devices, we introduce the use of Watts-Strogatz (WS) [3] network priors as the basis for hidden layer architectures in FL models, inspired by a similar approach proposed by [4] for centralized learning settings. This technique represents a form of neural network sparsification, which is done prior to the learning process. By doing so, network weights are pruned and rewired

SQAMIA 2024: Workshop on Software Quality, Analysis, Monitoring, Improvement, and Applications, September 9–11, 2024, Novi Sad, Serbia

[*]Corresponding author.

milic@dmi.uns.ac.rs (M. Ilić); svc@dmi.uns.ac.rs (M. Savić); mira@dmi.uns.ac.rs (M. Ivanović)
https://perun.pmf.uns.ac.rs/wiki/mihailo-ilic (M. Ilić); http://people.dmi.uns.ac.rs/~svc/ (M. Savić); https://perun.pmf.uns.ac.rs/wiki/mirjana-ivanovic (M. Ivanović)
0009-0005-4492-017X (M. Ilić); 0000-0003-1267-5411 (M. Savić); 0000-0003-1946-0384 (M. Ivanović)

immediately upon the initialization of a new model to be trained, thus reducing the number of weights of the model. This in turn introduces benefits such as reduced model size, less computations carried out during training, and the reduction of the size of messages being communicated between the FL server and the edge nodes.

In this paper we demonstrate that:

- by selecting optimal WS network priors, quality ML models can be obtained;
- the WS model can match the performance of regular (unpruned) fully connected models in FL in the same amount of training rounds conducted.

The rest of the paper is structured as follows. Section 2 covers related work done on FL optimization and neural network pruning. Watts-Strogatz network priors and the construction of hidden layer architectures from them is discussed in Section 3. Experimental results are showcased and discussed in Section 4, while concluding remarks and future work are given in Section 5.

2. Related Work

Optimizing FL is difficult as it has a wide range of applications, all of which can have their own unique challenges based on the setup of the environment. For instance, the number of participating edge nodes may vary from only a handful to hundreds of thousands of participants. The nature of the data (it being private or not) and the scale of data collection affect which are the main concerns in each scenario. In some situations privacy preservation is the main focus, while for others handling communication overhead is a top priority. Consequently, there are numerous ways of approaching the problem of optimizing FL.

One main approach to optimizing FL is the development of new aggregation algorithms. In FL, each edge node makes its own updates to the global model, which then need to be aggregated by the FL server. The most common way of doing this is by using FedAvg [5]. Research into algorithmic optimization is focused on creating new aggregation algorithms like FedProx and FedHybrid [6, 7]. Contrary to this, our approach is focused on the ML models themselves, and not the aggregation algorithm. The experiment presented in this paper uses FedAvg, but is not limited to it as it can also rely on different aggregation algorithms.

Edge nodes can make updates to the model either simultaneously (concurrently) or one after the other (incrementally). Comparisons on the performance of models obtained through different methods and optimal use of different updating schemes have been done in [1, 8, 9]. The sparse models we introduce to FL in this paper can be used with any of the updating strategies proposed in related work.

Communication efficiency can become a more prevalent issue once the number of participating edge nodes is large enough. This can have negative impacts on the system, like those caused by the server-side communication bottleneck. Solutions to this problem include either sending fewer messages, for instance by optimally selecting clients, or by quantizing the messages to reduce their size [10, 11, 12, 13, 14, 15]. The approach outlined in our paper belongs to the latter solution, as message size is directly reduced as a consequence of reducing the model size through sparsification.

By addressing how deep models used in FL are constructed, and by reducing the number of weights, model and message size can be reduced, in turn lowering the strain in communication. Furthermore, if this is achieved prior to model training, it can significantly reduce computational strain during learning. By introducing Watts-Strogatz graphs as network priors in deep learning, both of these objectives can be achieved. Our paper builds upon prior work done in centralized environments [16, 17, 4], where it has been shown that these models can match the performance of regular fully connected networks. We introduce this technique to FL and validate the approach, thus showing that it is a viable technique of optimizing FL.

3. Watts-Strogatz Network Priors

The rise of network science started with the study by Watts and Strogatz [3] on small-world networks. The authors examined the structure of three real-world complex networks from different domains (biological, technological and social) noticing that the analyzed networks possess the small-world property, i.e. small distances between randomly selected nodes [18]. At the same time, they exhibit drastically higher degree of local clustering than comparable Erdos-Renyi random graphs [19], where the degree of local clustering is defined as the probability that two nodes having a common neighbour are also neighbours [18]. This empirical observation was significant because the classic theory of random graphs cannot explain the presence of these two qualities together in one large and sparse graph.

To explain this phenomenon, Watts and Strogatz investigated characteristics of a random network model that interpolates between ring lattices and Erdos-Renyi random graphs. Ring lattices are regular graphs obtained by placing nodes on a ring and connecting each node to its k-nearest neighbors, $k/2$ on each side (k should be an even number). Ring lattices exhibit a large local clustering, but sparse ring lattices are not small-world graphs. Therefore, ring lattices have exactly opposite characteristics of local clustering and shortest paths compared to Erdos-Renyi random graphs. In the Watts-Strogatz (WS) model, we start from a ring lattice. Then, each link connecting n_i to a neighbor n_j, $i < j$, is rewired with a probability p to a randomly selected node sampled in a way to avoid parallel links and loops. For $p = 0$ the WS model generates ring lattices, while for $p = 1$ it produces pure random graphs. Watts and Strogatz showed that the rewiring procedure for $0.01 < p < 0.1$ produces graphs having at the same time the small-word property and a high degree of local clustering.

Algorithm 1: The Watts-Strogatz algorithm for sampling small-world random graphs.

```
input  : $N$ – the number of nodes
         $k$ – the number of nearest neighbors ($k \ll N$)
         $p$ – the rewiring probability for links
         random() – a function returning a pseudo-random number in $[0, 1)$
output : $G = (V, E)$ – Watts-Strogatz random graph

$V = \{1, 2, \ldots, N\}$
$E = \emptyset$

for $i = 1$ to $N$ do
    for $j = 1$ to $k/2$ do
        $d = i + j$
        if $d > N$ then
            $d = d - N$
        end
        $E = E \cup \{\{i, d\}\}$
    end
end

foreach $\{i, j\} \in E : i < j$ do
    if random() $\leq p$ then
        repeat
            $d = 1 + \lfloor \text{random}() \cdot N \rfloor$
        until $d \neq i \wedge \{i, d\} \notin E$
        $E = E \setminus \{\{i, j\}\}$
        $E = E \cup \{\{i, d\}\}$
    end
end
```

3.1. Building Hidden Layers from WS Network Priors

The approach outlined in Algorithm 1 yields graph structures which can then be transformed into hidden layers of an artificial neural network. An algorithm for this purpose is described in [16], where the undirected WS graph prior is first transformed into a directed graph, as seen in Figure 1a. Following this, all of the vertices are placed into layers using a recursive function which guarantees the correct ordering of layers, having each node be in a deeper layer than its predecessors. Vertices with an in-degree of 0 represent the nodes in the first layer, while those with an out-degree of 0 are the nodes which are to be placed in the final layer of the network. Based on the input parameters provided to Algorithm 1, different configurations in terms of sparsity and skip connections can be observed, an example is in Figure 1b. This network architecture can then be used for any problem set only by placing additional input and output layers to the network.

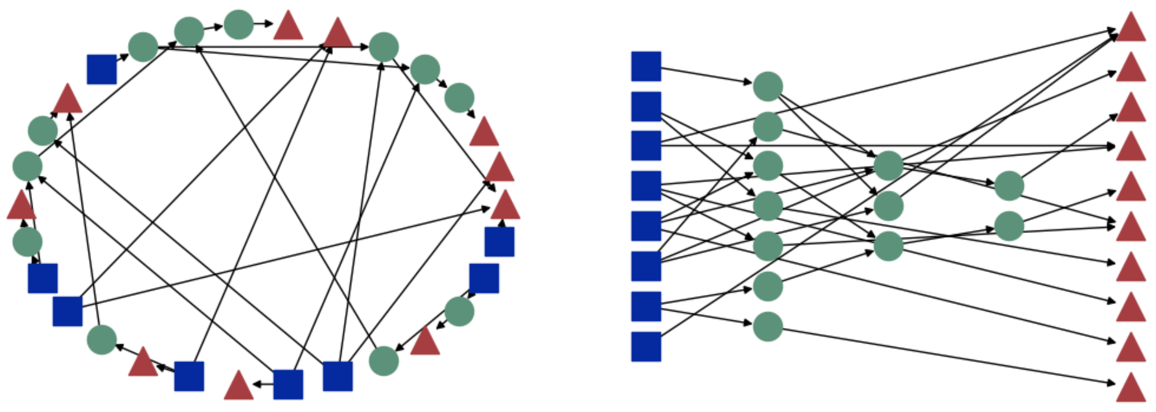

(a) Directed WS graph. (b) WS-based hidden layer architecture.

Figure 1: The transformation of a Watts-Strogatz graph into a hidden layer architecture. The input parameters for the WS model are $N = 30, k = 2, p = 0.5$. Blue squares have an in-degree equal to 0 and are placed in the first layer, red triangles have an out-degree of 0 and are placed in the last layer, while all other nodes (green circles) are in various layers in between.

Different inputs of N, k, and p yield different graph structures, some of which are more suited to be baseline architectures for artificial neural networks. For instance, by providing lower values for the parameter p, random rewiring of the edges is less common, which in turn leads to the layer transformation method described in [16] constructing deeper but narrower hidden layer structures, which may be more susceptible to the vanishing gradient problem. An example can be seen in Figure 2, where the network was generated using $p = 0.1$, as opposed to what can be observed in Figure 1 when p was set to 0.5.

The use of WS network priors in deep learning has been shown to give reliable models [4, 17] in centralized learning scenarios. By constructing hidden layers in such a manner, network sparsification is done implicitly prior to the training process. Network pruning is a process which is usually carried out after model training, by applying some sort of importance metric to each weight in the network. Furthermore, WS based networks introduce skip connections naturally to the network, which improves the flexibility and robustness of the models.

In this paper we propose the use of WS network priors in FL. Computability and communication challenges in resource constrained federated environments can directly benefit from this approach, as it provides smaller, yet still effective, models, thus addressing computability and communication costs.

4. Results & Discussion

The aim of the experiment in this paper was to test the viability of using WS-based neural networks in FL environments. We created a series of WS network priors based on the technique outlined in Section 3.

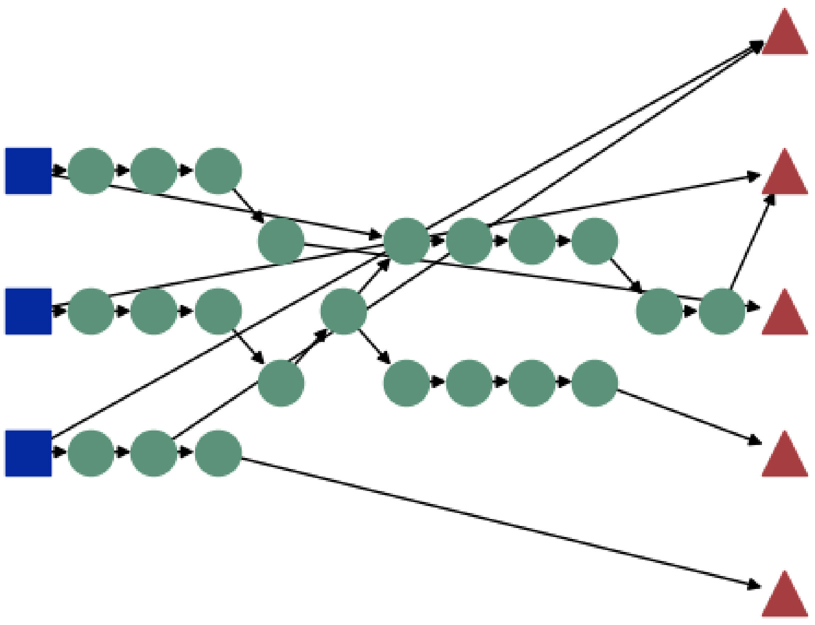


Figure 2: A hidden layer architecture obtained by using $N = 30, k = 2, p = 0.1$ as inputs to the WS network prior generation algorithm.

For each of these hidden layer structures, a pair of models were constructed. The first model is the WS model directly defined through this method, while the second model is a regular, fully connected, model constructed by taking the layers from the WS graph and simply connecting all neurons in neighbouring layers together.

4.1. The Dataset

In the experiment one dataset from the LEAF collection of FL benchmark datasets [20] was used. LEAF was chosen as it captures the non-IID properties of datasets similar to real-world applications of FL. It provides a method of synthetic data generation for multiclass classification. So, a synthetic dataset was generated, using the tools provided within LEAF, having 60 input parameters and 5 classes, and with 250 edge nodes in total in the system. A histogram depicting the number of data points available to edge nodes can be seen in Figure 3. A vast majority of the edge nodes have less than 50 data points available, while only a handful have more than 100.

A single layer perceptron was trained [20] on a synthetically generated dataset on the same data format with 1000 edge nodes available, as opposed to the 250 edge nodes in our experiments. Their model achieved a weighted (per data point) accuracy of 71.89%, which can be used as a reference when comparing model performance of WS models.

4.2. Results

The experiment consisted of 250 edge nodes in the federated environment, all with varying sizes of local datasets. Training lasted for 75 rounds, where in each one 10 random edge nodes were selected to provide their updates to the global model. Local training lasted 1 epoch for each client with a batch size of 5 and a learning rate (LR) of 0.01 for stochastic gradient descent.

In total 3 random WS architectures were generated, consisting of 150 nodes (neurons), with $k = 2$ and $p = 0.5$. From this, 3 pairs of models were created having 1 WS model and 1 fully connected model based on the same architecture. Table 1 contains the number of trainable parameters for all 6 models (3 pairs of regular and WS models). It is evident that the WS models have far less trainable parameters compared to the regular networks.

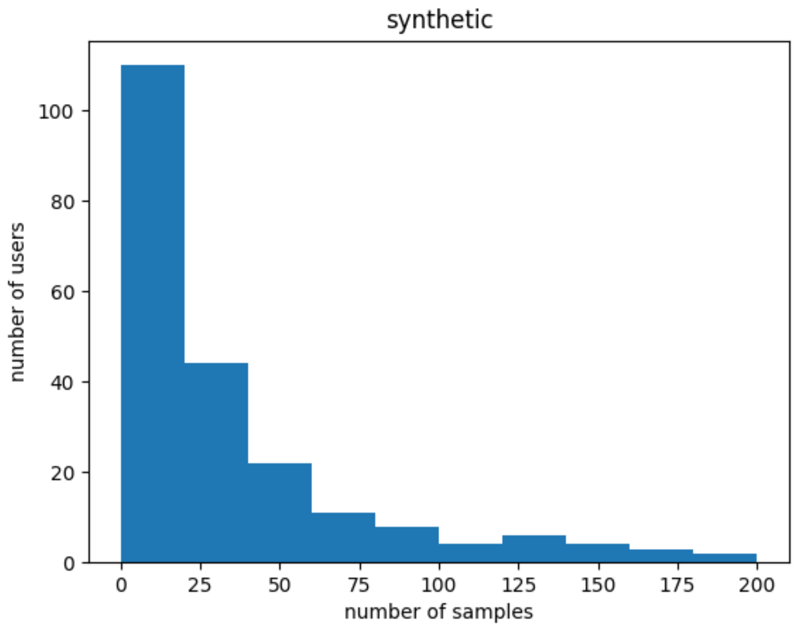


Figure 3: A histogram depicting data availability across the federated environment. The X axis denotes the number of samples, while the Y axis denotes the number of data points available at the edge nodes.

Table 1
The number of trainable parameters for each pair of regular and WS models created from the 3 hidden layer structures created for the experiment.

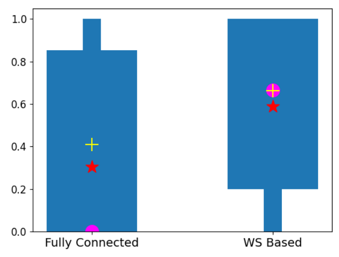
Structure ID	Regular Model # Params	WS Model # Params
1	3644	2275
2	3671	2445
3	3650	2455

After 75 global training steps, all models were evaluated on the entirety of the client pool, that is on all available edge nodes. Each edge node split its local data 80:20 into training and testing data. When evaluating federated models, in this case their accuracy, it is vital to observe both *per client* and *per data point* metrics, i.e. their *average accuracy* across edge nodes and their *weighted average accuracy* across edge nodes. The weighted average is obtained when the size of training data available at each edge node is also taken into account.

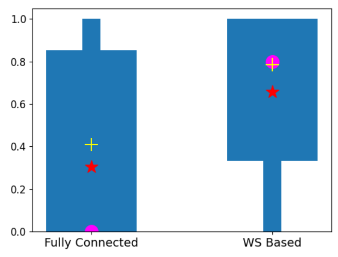
Figures 4 and 5 display a series of candlestick charts representing model accuracies for each of the 3 pairs of models side by side. The wider parts of the candlesticks indicate the 25th and 75th percentile range, while the narrower parts extend to the 10th and 90th percentiles. Additionally, 3 markers on each candlestick indicate specific accuracy metrics: a red star for the mean accuracy, a magenta circle for median accuracy, and a yellow plus for the weighted (per data point) accuracy.

Figure 4 takes into account all of the edge nodes available in the client pool. The 10th percentile accuracy for all models here is 0%, a consequence of having edge nodes with only 3 data points available, which is the minimum any edge node can have in this setup. By looking at all 3 pairs of candlesticks, it is evident that the WS based sparse network achieved better results across the federated environment. As the 25th percentile is higher for the WS model compared to the regular network in all cases, we can conclude that it achieves better *per client* performance. The weighted average score of WS models is also significantly higher for all 3 pairs.

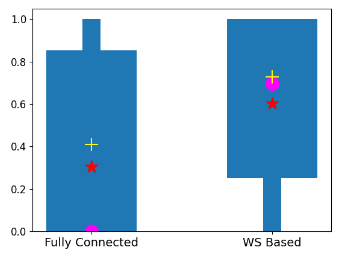
Figure 5 excludes edge nodes with fewer than 30 data points. These edge nodes provide a lot less data, and consequently have far less impact on the updates to the global model. By observing the candlesticks



(a) Architecture 1 metrics.

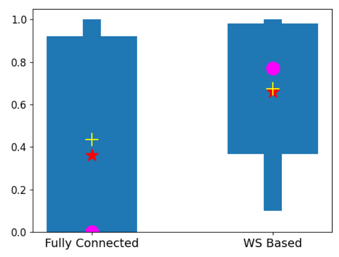


(b) Architecture 2 metrics.

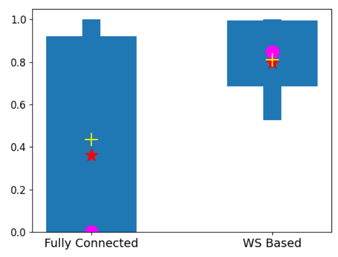


(c) Architecture 3 metrics.

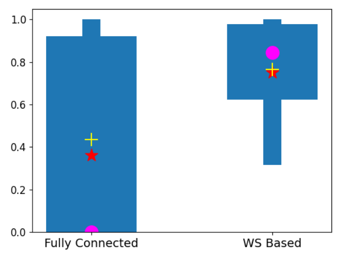
Figure 4: Accuracy metrics for all 3 architectures across the entire edge node pool. The markers represent the following: mean (red star), median (magenta circle), weighted average (yellow cross).



(a) Architecture 1 metrics.



(b) Architecture 2 metrics.



(c) Architecture 3 metrics.

Figure 5: Accuracy metrics for all 3 architectures across edge nodes with at least 30 data points available. The markers represent the following: mean (red star), median (magenta circle), weighted average (yellow cross).

we can conclude that the WS models have far better accuracy results across the board, which holds especially for architecture 2. On the other hand, the regular models did see minor improvement, however, not enough.

The WS models achieved better accuracy across the board. This can be explained by the impact learning rate has on the weights of the models. Since all WS models have roughly 2 thirds of trainable parameters the fully connected networks have, backpropagation has a more significant impact on the weights in the WS model compared to those in the fully connected one, since the impact there is more spread out between the weights. Future work will see an adjustment here, where the LR will be scaled proportionately to the number of trainable parameters.

Nevertheless, an important finding is that the WS models can indeed match, and even surpass, the performance of the regular, unpruned, networks. In the same number of global learning rounds, far less computations were needed to achieve these results. This goes in favour of using WS network priors in federated setups, as it reduces memory consumption, computational workload, and message size.

5. Conclusion & Future Work

Optimization of FL in massively distributed environments can be viewed in many different ways, depending on the specific criterion to be enhanced. For instance, optimizing model performance by means of clustering and personalized FL, optimizing communication through message quantization, reducing communication costs through user sampling, etc.

This paper presents an approach to optimized resource use by introducing network science concepts to FL. By using Watts-Strogatz graphs as a baseline for hidden layer architectures in artificial neural networks, the model size can be reduced immediately upon creation of the model, prior to training. This presents an alternative to network pruning, a process of weight elimination from neural networks which is done usually following training. The approach outlined in this paper reduces the size of the model, the number of computations needed during training, and the size of the messages being communicated between the cloud and edge. It brings added benefits to memory consumption, computation cost, and communication efficiency.

This paper demonstrates the viability of using WS network priors in FL settings, and outlines the benefits of such an approach. Future work will include an expanded dataset collection with different types of input data (e.g. images), and different inference tasks (such as regression). Furthermore, it is vital to provide a more detailed analysis on the relationship between specific graph properties and their effectiveness as hidden layer structures in artificial neural networks. Finally, WS as a method of sparsification should be tested against other sparsification methods, for instance L1 and L2 regularization.

Acknowledgments

Mihailo Ilić, Miloš Savić, and Mirjana Ivanović gratefully acknowledge the financial support of the Ministry of Science, Technological Development and Innovation of the Republic of Serbia (Grants No. 451-03-66/2024-03/200125 & 451-03-65/2024-03/200125).

Mirjana Ivanović gratefully acknowledge the partial supporte by CERCIRAS - "Connecting Education and Research Communities for an Innovative Resource Aware Society" COST Action CA19135 funded by COST Association.

References

[1] M. Ilić, M. Ivanović, V. Kurbalija, A. Valachis, Towards optimal learning: Investigating the impact of different model updating strategies in federated learning, Expert Systems with Applications 249 (2024) 123553.

[2] M. Savić, V. Kurbalija, M. Ilić, M. Ivanović, D. Jakovetić, A. Valachis, S. Autexier, J. Rust, T. Kosmidis, The application of machine learning techniques in prediction of quality of life features for cancer patients, Computer Science and Information Systems 20 (2023) 381–404.

[3] D. J. Watts, S. H. Strogatz, Collective dynamics of'small-world'networks, Nature 393 (1998) 409–10. doi:10.1038/30918.

[4] T. Traub, M. Nashouqu, L. Gulyás, Efficient sparse networks from watts-strogatz network priors, in: International Conference on Computational Collective Intelligence, Springer, 2023, pp. 163–175.

[5] B. McMahan, E. Moore, D. Ramage, S. Hampson, B. A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: Artificial intelligence and statistics, PMLR, 2017, pp. 1273–1282.

[6] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, V. Smith, Federated optimization in heterogeneous networks, Proceedings of Machine learning and systems 2 (2020) 429–450.

[7] X. Niu, E. Wei, Fedhybrid: A hybrid federated optimization method for heterogeneous clients, IEEE Transactions on Signal Processing 71 (2023) 150–163.

[8] M. J. Sheller, G. A. Reina, B. Edwards, J. Martin, S. Bakas, Multi-institutional deep learning modeling without sharing patient data: A feasibility study on brain tumor segmentation, in: Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries: 4th International Workshop, BrainLes 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 16, 2018, Revised Selected Papers, Part I 4, Springer, 2019, pp. 92–104.

[9] M. J. Sheller, B. Edwards, G. A. Reina, J. Martin, S. Pati, A. Kotrotsou, M. Milchenko, W. Xu, D. Marcus, R. R. Colen, et al., Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data, Scientific reports 10 (2020) 1–12.

[10] D. Alistarh, D. Grubic, J. Li, R. Tomioka, M. Vojnovic, Qsgd: Communication-efficient sgd via gradient quantization and encoding, Advances in neural information processing systems 30 (2017).

[11] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, R. Pedarsani, Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2020, pp. 2021–2031.

[12] A. Koloskova, S. Stich, M. Jaggi, Decentralized stochastic optimization and gossip algorithms with compressed communication, in: International Conference on Machine Learning, PMLR, 2019, pp. 3478–3487.

[13] Y. J. Cho, J. Wang, G. Joshi, Client selection in federated learning: Convergence analysis and power-of-choice selection strategies, arXiv preprint arXiv:2010.01243 (2020).

[14] W. Chen, S. Horvath, P. Richtarik, Optimal client sampling for federated learning, arXiv preprint arXiv:2010.13723 (2020).

[15] M. Ribero, H. Vikalo, Communication-efficient federated learning via optimal client sampling, arXiv preprint arXiv:2007.15197 (2020).

[16] J. Stier, M. Granitzer, Structural analysis of sparse neural networks, Procedia Computer Science 159 (2019) 107–116.

[17] M. B. Amor, J. Stier, M. Granitzer, Correlation analysis between the robustness of sparse neural networks and their random hidden structural priors, Procedia Computer Science 192 (2021) 4073–4082.

[18] M. Savić, M. Ivanović, L. C. Jain, Fundamentals of Complex Network Analysis, Springer International Publishing, Cham, 2019, pp. 17–56. doi:10.1007/978-3-319-91196-0_2.

[19] P. Erdős, A. Rényi, On the evolution of random graphs, Publications of the Mathematical Institute of the Hungarian Academy of Sciences 5 (1960) 17–61.

[20] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konečný, H. B. McMahan, V. Smith, A. Talwalkar, Leaf: A benchmark for federated settings, 2019. URL: https://arxiv.org/abs/1812.01097. arXiv:1812.01097.