FedPruNet: Federated Learning Using Pruning Neural Network

Gowtham L
Dept. of Computer Science and Engineering
National Institute of Technology Karnataka, Surathkal
Mangaluru, India
gowtham.203cs001@nitk.edu.in

Annappa B
Dept. of Computer Science and Engineering
National Institute of Technology Karnataka, Surathkal
Mangaluru, India
annappa@ieee.org

Sachin D N
Dept. of Computer Science and Engineering
National Institute of Technology Karnataka, Surathkal
Mangaluru, India
sachindn.207cs004@nitk.edu.in

2022 IEEE Region 10 Symposium (TENSYMP) | 978-1-6654-6658-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/TENSYMP54529.2022.9864565

***Abstract*—Federated Learning (FL) is a distributed form of training the machine learning and deep learning models on the data spread over heterogeneous edge devices. The global model at the server learns by aggregating local models sent by the edge devices, maintaining data privacy, and lowering communication costs by just communicating model updates. The edge devices on which the model gets trained usually will have limitations towards power resource, storage, computations to train the model. This paper address the computation overhead issue on the edge devices by presenting a new method named *FedPruNet*, which trains the model in edge devices using the neural network model pruning method. The proposed method successfully reduced the computation overhead on edge devices by pruning the model. Experimental results show that for the fixed number of communication rounds, the model parameters are pruned up to 41.35% and 65% on MNIST and CIFAR-10 datasets, respectively, without compromising the accuracy compared to training FL edge devices without pruning.**

***Index Terms*—Federated Learning, edge computing, model pruning, deep learning, neural networks**

I. Introduction

Traditional training methods of machine learning and deep learning model involve lots of data stored at the centralized server. Initially, the data is sent to the server, stored, and used to learn the models. However, due to an increase in data privacy concerns and new privacy policies like General Data Protection Regulation (GDPR) [1], Federated Learning (FL) came as a new solution. Researchers of Google initially proposed FL in the year 2017 [2]. In the FL approach, only the model updates like the neural network parameters are communicated, which eventually reduces the required data transfer between the clients and the server, improving privacy as all the computations are performed locally at the edge side using their data. From the inception of FL, various applications like Google keyboard next word prediction [3], Apple ios [4] and google pixel phone [5] are also using FL for quick keyboard type and voice classification. FL is also used for predicting financial risk in insurance [4], the discovery of pharmaceuticals products [4], predictive maintenance and smart manufacturing [4], etc.

A. Federated Learning Overview

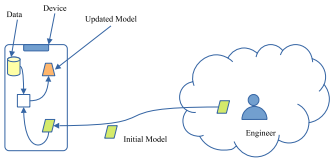
The initial client selection happens by sampling a set of clients from all the clients that meet the requirements. The selected clients receive the initial model and train the model, and compute an update to the model (Figure 1a). The updated local models from the clients are sent back to the server. The global model is generated by aggregating the local model updates using FedAvg [2] at the server (Figure 1b). The updated combined global model is broadcasted to the next set of clients for the training (Figure 1c) and is repeated for the next set of rounds till the model converges (Figure 1d).

B. Challenges in Federated Learning

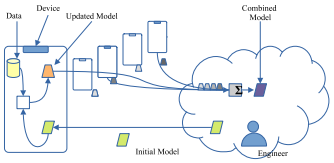
Several challenges exist in FL. There are several open research problems to solve under each, and these challenges are categorized into the following domain:

- **Expensive Communication:** FL System is made up of various edge devices like IoT devices, smartphones, etc. The communication in these networks is slower as there is a difference in the network upload and download speed. Due to this, the communication expenses of the model are high. [6]
- **Heterogeneous systems:** The devices present in FL vary with their computational, storage, and communication efficiency due to differences in their hardware components and network speed (3G, 4G, 5G, Wi-Fi) and their battery levels [6]. Hence these devices are unreliable during training due to connectivity and energy constraints. Thus developing the system to be reliable during training and developing the methods to handle these resource-constrained devices during training in FL is an open research problem.

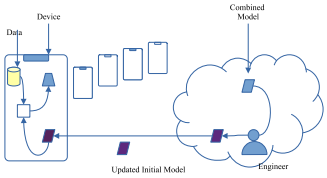
978-1-6654-6658-5/22/$31.00 ©2022 IEEE



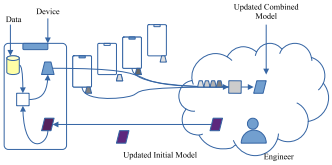
(a) Initial model distribution to clients (Step 1)



(b) Updated models aggregation (Step 2)



(c) Aggregated combined model distribution (Step 3)



(d) Updated models aggregation (Step 4)

Fig. 1: Federated Learning Overview

Data are being generated every day by lots of edge devices like IoT devices, autonomous vehicles, etc. The processing and decision should be taken in real-time using the live data. In FL, the models are frequently communicated between clients and the server. These models are trained in computationally weak edge devices. Some existing methods discussed in the next section solve the problem by training the model in parallel, which reduces the number of computations, but there is a need for many clients to participate during the training process. Some existing methods minimize communication rounds while training, thereby reducing the computations. However, the computations per client are not reduced.

This paper proposes a new method *FedPruNet* to overcome the above challenges. The technique discussed in the subsequent section does not require training the model until convergence before pruning. The channels are pruned while training the model after every specific set of rounds, unlike the traditional approach. After every prefixed set of communication rounds, a specified percentage of the network's least relevant channels are pruned using a feature relevance score. The relevance of the channel is determined before pruning. The feature relevance score is a data-driven measure that calculates the node's importance of each channel in the network [7]. Taking the scores of all the nodes of a specific feature map layer and averaging them yields the feature relevance score of a particular feature map f_i at a layer l. The derived feature relevance score identifies and prunes the network's least essential channels.

The contributions of this paper are:

- Proposed a new way of training the model in FL using the neural network model pruning method.
- Use of static and dynamic approach for pruning the model. In the static method, weights are pruned based on the threshold value, and in the dynamic method, the model is pruned by calculating the feature relevance score of the channel.
- It is shown that, with proposed approach, the model can be trained with less overhead on the edge devices without sharing the data and with a lesser number of model parameters without compromising the accuracy compared to the normal FL method.

The remaining part of this paper is organized as follows. Section II discusses the existing works related to reducing the computation overhead. Section III includes details about the proposed methodology. Section IV includes implementation details and results. Finally, Section V concludes the work with future open problems.

II. Existing Works

In FL, the server and the clients communicate frequently and involve sending and receiving global model and local models from server to client and client to server, respectively. These models include the parameters like weights and their updates. The deep learning model involves lots of parameters. The updates in CNN contain millions of parameters [8]. These highly parameterized updates may lead to high communication costs and computation overhead on resource-constrained edge devices.

Several methods exist which reduce the communication and computational cost, one of which is performing deep learning model training at the edge side. McMahan et al. [2] proposed two ways of reducing the communication cost. First, training in parallel, where multiple clients are selected and trained in parallel, and second, increasing computation per client, where more iterations are performed locally before communicating the model for global aggregation. Two major algorithms in the study are Federated Averaging (FedAvg) [2] and Federated SGD (FedSGD) [2]. In FedSGD, one pass of training occurs. In FedAvg, more computations are carried out on the local client-side. Yao et al. [9] proposed an extended version of the typical FL process, which reduces communication by increasing computation that follows the two-stream model approach. The global model is taken as a reference model in every round and compared with the local model. The model will learn from the local data and the knowledge shared by the other clients through the global model [10]. Jakub et al. [11] proposed a method for model compressing where it discusses two kinds of updates: structured updates and sketched updates. The structured model updates will have predefined low rank and random mask structures. The product of two matrices is used to represent the low-rank matrix. One matrix is created at random and kept static, whereas another is sent to the server for optimization. The matrix is represented as a sparse matrix in the random mask approach, and non-zero entries are sent to the server. Caldas et al. [12] proposed a new system which is an extension of the previous method. This method follows lossy compression and dropout, which mitigates the cost of server-to-client communication. Qiao et al. [13] proposed an

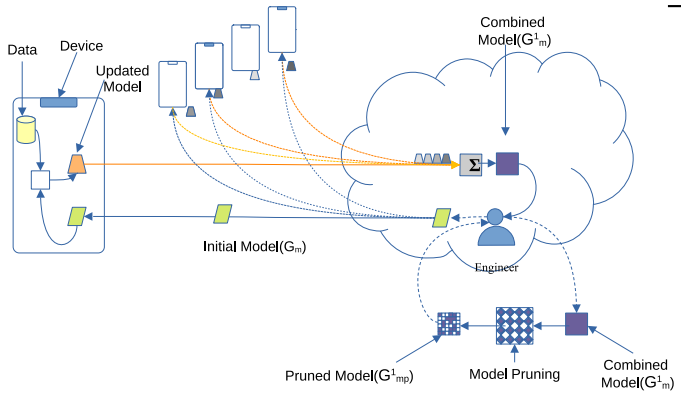


Fig. 2: Proposed System

Algorithm 1: FedPruNet

At Server Side:
set weights w_0
for *every round r = 1,2 ...,r* **do**
  $S_t \leftarrow$ randomly select m clients
  if *r % 20 and r* $\leq$ *200* **then**
    Using weights w calculate feature relevance score of the neural network
    Prune filters with least relevance score over all the layers
  end
  for *every client s* $\in S_t$ **do**
    $w^s_{t+1} \leftarrow$ UpdateClientModel(w_t)
  end
  $w_{t+1} \leftarrow$ FedAvg(w^s_{t+1})
end
At Client Side:
UpdateClientModel(w_l)**:**
$B \leftarrow$ divide client's dataset in batches
for *every epoch e* $\in E$ **do**
  for *every batch b* $\in B$ **do**
    $w_l \leftarrow w_l$ - $\eta \nabla\, l(w_l; b)$
  end
end
return w_l to server

extended version of the model compression technique. The model is compressed at both client and server-side before communicating it over the network. Yang et al. [14] proposed a technique that trains the FL model in a ring-like structure. The model updated from the client data gets forwarded to the nearest client instead of sending it to the server, and the final single updated model is sent to the server. Diao et al. [15] proposed a HeteroFL framework that decreases the communication and computation cost by training local models generated from a single global model based on edge device computation capabilities. Jiang et al. [16] presented PruneFL, which is based on the adaptive and distributed way of pruning. Clients are randomly selected and pruned using local data and followed by adaptive pruning. However, the initial model can be biased towards selected clients, and the model may not load into the memory of the resource-constrained devices.

Observations from the existing approaches are that some tried to reduce the communication between client and server but did not reduce computation overhead on the client. FL is based on edge training, which happens on the client side. The devices on the edge side are resource-constrained (mobile devices, IoT devices, etc.) and will have less computational power than a cloud server.

III. Methodology

Neural network pruning is one of the widely used techniques for reducing the model size [17]. This strategy minimizes the overall complexity of neural networks by reducing the number of parameters in the network without impacting the model's accuracy. Two types of pruning are performed, static pruning and dynamic pruning. In static pruning, the neural network's weights that are less than the threshold value are pruned [18] and continue retraining the model in the next round of communication. In the dynamic approach, the relevance-based gradual channel pruning [19] is adopted in the proposed edge-based model training in the federated learning system. Pruning is performed at the server-side frequently after every specific set of rounds.

The proposed system architecture is as shown in Figure 2. Initially, an FL system is constructed by taking a different set of clients. Random clients are chosen from the set of available ones in a particular round and sent the initial model. These clients train the model and send back the updated model to the server. The local updated model are then aggregated using FedAvg [2] to generate an updated global model. After generating the global model, The pruning method is applied to generate the pruned global model. The relevance-based gradual channel pruning [19] uses a feature relevance score and identifies the importance of the network channels. The contribution of each node to the final prediction of the model is determined using the layer-wise relevance propagation technique introduced in [7]. For a given input data, taking activation and weight values of the network, every node in the network gets a relevance score. The output node relevance score is determined by the true label of the given instance. The relevance score is set to 1 if the output node evaluates to the true class and the remaining nodes are set to 0. The relevance scores are communicated back in the network using the $\alpha - \beta$ decomposition rule [20]. Averaging the scores of all the nodes at a particular layer l of the feature map, the feature relevance score of each feature map f_i is obtained. This feature relevance score determines the least important channels in the network and is pruned. This method is applied after every predefined training round

between the clients and the server. This pruned global model is sent back to random clients containing heterogeneous data from the pool of clients registered for the next training rounds. Algorithm 1 shows the steps of the proposed methodology. The weights get initialized when an initial model is built on the server-side. In every round 'r', a random set of clients 'S_t' is selected from the available 'm' clients. Pruning is performed after every 20^{th} round and stops at the 180^{th} round. For every client 's' selected from the set 'S_t', the global model (initial weights) is sent. On the client-side, model training happens by dividing the datasets into batches 'b'. For the local epochs *E=5*, the gradients are calculated using stochastic gradient descent (SGD) [21], and the updated model is sent to the server. The server then performs the model aggregation using the FedAvg [2], and the process is repeated until the conditions are satisfied.

IV. Experiments

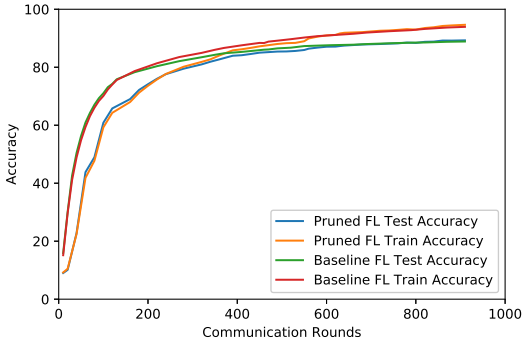
This section discusses the experimental setup and results in terms of accuracy, pruning percentage, and reduction of Floating Point Operations (FLOPs) compared to the proposed methodology and FL without pruning. The accuracy, pruned rate, the number of non-zero parameters, and FLOPs of normal FL training and proposed pruned FL training approach are reported.

A. Implementation Details

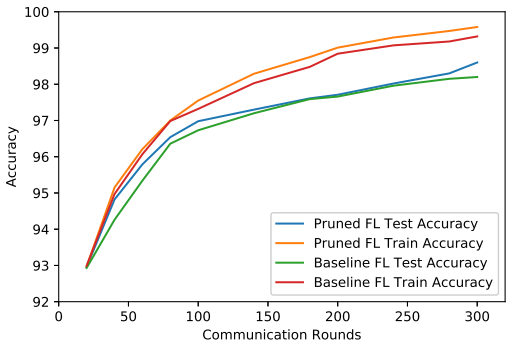
The proposed method *FedPruNet* is built using the PyTorch [22] framework. Experiments are carried out on MNIST [23] and CIFAR-10 [24] datasets. LeNet 300 model and VGG16 are used on MNIST and CIFAR-10, respectively. The FL system is set up by simulating 100 clients sequentially on a single device for experimental purposes and dividing the dataset equally. CIFAR-10 dataset contained 50000 training images. Each client includes 500 random images. 1% (10 clients of 100) of clients are randomly chosen in every training round and trained the model on their local data. While training the model to reduce the computation overhead, the model is trained by fixing the number of epochs *E=5* per client. Experimented by choosing *E=5* per client because the clients in FL system are usually edge devices with limited resources and less computation power. The model is trained successfully with lesser epochs per client. The proposed method shows that the accuracy is achieved on par with the baseline models by decreasing the computation overhead on clients and the number of parameters in the model.

B. Results

For the CIFAR-10 dataset, the model is pruned for every 20 communication rounds by calculating the relevance score. The pruning step is applied at 20 intervals and has paused at the 180^{th} round, as pruning further rounds will have a convergence issue [17]. With the pruning rate of 56.43% at the end of 180 FL rounds, the test and train accuracy was 66.92% and 65.57%, respectively. With the pruned rate of 56.43%, the model is



(a) Accuracy vs communication rounds in CIFAR-10



(b) Accuracy vs communication rounds in MNIST

Fig. 3: Train and Test accuracy at different stage of FL rounds with pruning and without pruning

trained until the 910 communication round. At the end of the 910 communication round with the pruning rate of 64.99%, the test-train accuracy before and after pruning was 89.29%-94.67% and 89.28%-94.64%, respectively. Further training is stopped as accuracy before and after pruning at this pruned stage was the same. For comparison, the model is trained with the same data without pruning the model in FL way for 910 communication rounds. The test and train accuracy of 88.08% and 93.23%, respectively, is finally achieved, as shown in Figure 3a. Initially, the model contained 14986698 non-zero parameters. The non-zero elements in the normal FL way of training had all the initial parameters (14986698). In contrast, the pruned model at the pruning rate of 64.99% had 5245362 parameters, and the other remaining parameters are nullified (parameters with zero values). The floating-point operations (FLOPs) for normal baseline FL measured 3.14×10^8 FLOPs, whereas the pruned FL had 1.10×10^8 FLOPs. From Figure 4a the parameters during the initial and final stages of the proposed pruned FL vs. normal FL way of training shows that the proposed approach achieved better accuracy than the normal FL way of training. The computation overhead is minimized while training the model in FL systems by reducing the parameters

TABLE I: Performance of proposed approach with different pruning rate on CIFAR-10 dataset

Pruned Percentage	Communication Rounds	Non Zero Parameters	Zero Parameters	Pruned Rate	Accuracy before Pruning (Test, Train)	Accuracy After Pruning (Test, Train)
13.78	20	13027922	1958776	1.15x	10.11, 10.47	10.78, 10.84
19.54	40	12064006	2922692	1.24x	22.80, 22.50	21.75, 21.12
25.45	60	11175320	3811378	1.34x	43.82, 41.85	41.12, 39.94
31.73	80	10234496	4752202	1.46x	53.24, 51.48	48.96, 47.76
38.11	100	9279290	5707408	1.61x	60.84, 59.26	57.47, 54.93
44.55	120	8314398	6672300	1.80x	65.80, 64.28	61.95, 60.10
47.65	140	7844596	7142102	1.91x	68.92, 66.83	63.72, 62.40
50.83	160	7372789	7613909	2.03x	69.02, 68.01	65.67, 63.93
56.43	180	6534141	8452557	2.29x	72.17, 71.23	66.92, 65.57
64.99	910	**5245362**	**9741336**	**2.85x**	**89.29, 94.67**	**89.28, 94.64**

TABLE II: Performance of proposed approach with different pruning rate on MNIST dataset

Pruned Percentage	Communication Rounds	Non Zero Parameters	Zero Parameters	Pruned Rate	Accuracy before Pruning (Test, Train)	Accuracy After Pruning (Test, Train)
23.53	20	203877	62733	1.30x	93.37, 93.36	92.96, 92.97
31.68	40	182137	84473	1.46x	94.66, 94.96	94.82, 95.15
33.21	60	178081	88529	1.50x	95.64, 96.07	95.79, 96.21
34.60	80	174362	92248	1.52x	96.41, 96.91	96.54, 97.00
35.55	100	171822	94788	1.55x	96.88, 97.47	96.98, 97.55
37.18	120	167513	99097	1.59x	97.30, 98.18	97.30, 98.29
38.40	140	164227	102383	1.62x	97.60, 98.69	97.61, 98.75
39.33	160	161764	104846	1.65x	97.95, 98.97	97.71, 99.01
40.21	180	159399	107211	1.67x	98.16, 99.25	98.02, 99.29
41.35	300	**156368**	**110242**	**1.71x**	**98.62, 99.60**	**98.60, 99.58**

TABLE III: Comparison of proposed pruned FL approach with baseline FL training

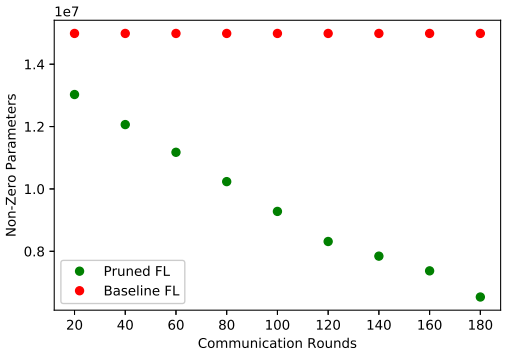
Dataset	Model	Communication Rounds	Accuracy of Baseline FL (Test, Train)	Accuracy of Pruned FL (Test, Train)	Parameters Alive in Baseline FL	Parameters Alive in Pruned FL	Percentage of Parameters Reduction	FLOPs Baseline FL	FLOPs Pruned FL
CIFAR	VGG16	910	88.08, 93.23	**89.28, 94.64**	14986698	**5245362**	**64.99**	3.14E+08	**1.10E+08**
MNIST	LeNet300	300	98.20, 99.32	**98.60, 99.58**	266610	**156368**	**41.35**	3.26E+05	**1.91E+05**

by 2.85 times the initial unpruned model. Table II and III summarize the performance of the model at different pruning rates and comparison results.

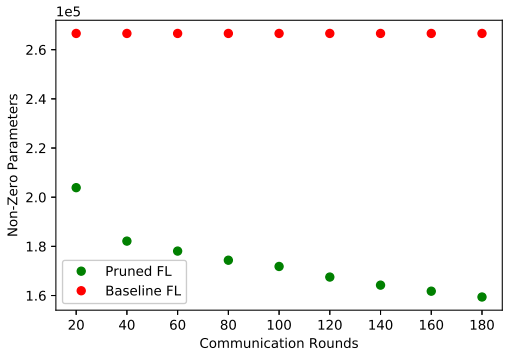
For the MNIST dataset, the pruning step is applied at 20 intervals and has paused at the 180^{th} round, as pruning further rounds will have a convergence issue [17]. With the pruning rate of 40.21% at the end of 180 FL rounds, the test-train accuracy before and after pruning was 98.02% and 99.29%, respectively. With the pruned rate of 40.21%, the model is trained until the 300 communication round. At the end of the 300 communication round with the pruning rate of 41.35%, the test-train accuracy before and after pruning was 98.62%-99.60% and 98.60%-99.58%, respectively. Further training is stopped as accuracy before and after pruning at this pruned stage was the same. For comparison, the model is trained with the same data without pruning the model in FL way for 300 communication rounds and achieved a test and train accuracy of 98.20% and 99.32%, respectively, as shown in Figure 3b. Figure 4b shows the parameters during the initial and final stages of the proposed pruned FL vs. normal FL way of training. The initial untrained model contained 266610 parameters, and the model trained by the normal FL way had all the parameters of the initial model, but the proposed approach had 156368 parameters at the pruning rate of 41.35%. The floating-point operations (FLOPs) for normal baseline FL measured 3.26×10^5 FLOPs, whereas the pruned FL had 1.91×10^5 FLOPs. The computation overhead is minimized while training the model in FL systems by reducing the parameters by 1.71 times the initial unpruned model. Table II and III summarize the performance of the model at different pruning rates and comparison results.

V. Conclusion

Edge devices in FL, such as mobile phones, IoT devices, and so on, are often resource-constrained and lack computing capacity compared to CPU and GPU servers. As a result, training the model locally in FL using these devices is time-consuming. This paper described the innovative way *FedPruNet*



(a) Non-Zero parameters vs communication rounds in CIFAR-10



(b) Non-Zero parameters vs communication rounds in MNIST

Fig. 4: Variation in number of parameters at different communication round in FL without pruning and FL with pruning

trains the model in the FL system by pruning the neural network based on the feature relevance score in this study. Results demonstrate the model parameters are reduced by up to 65% and 41.35% in CIFAR-10 and MNIST, respectively, without sacrificing accuracy. The results show that the proposed strategy outperformed the baseline method of training the model in the FL system without pruning. The breadth of study in FL is expanding, and there are many open research topics to answer. *FedPruNet* successfully reduced the computation overhead by pruning the parameters not required for matrix multiplication while training the model. In the current work, the edge device's computation cost in actual device settings is not explicitly measured. Evaluation of the computation cost in the real device setting is taken as future work. Also, the number of communication rounds in the proposed system is nearly the same. Future work involves developing an innovative solution to cut down the communication costs by minimizing the number of model updates between clients and the server.

REFERENCES

[1] J. P. Albrecht, "How the gdpr will change the world," *European Data Protection Law Review*, vol. 2, pp. 287–289, 2016.

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," *Artificial intelligence and statistics Proceedings of Machine Learning Research*, pp. 1273–1282, 2017.

[3] A. Hard, K. Rao, R. Mathews, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *ArXiv*, vol. abs/1811.03604, 2018.

[4] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, and K. Bonawit et al., "Advances and open problems in federated learning," *Found. Trends Mach. Learn.*, vol. 14, pp. 1–210, 2021.

[5] S. I. Ramaswamy, R. Mathews, K. Rao, and F. Beaufays, "Federated learning for emoji prediction in a mobile keyboard," *ArXiv*, vol. abs/1906.04329, 2019.

[6] T. Li, A. K. Sahu, A. S. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, pp. 50–60, 2020.

[7] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PLoS ONE*, vol. 10, 2015.

[8] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *ArXiv*, vol. abs/1511.08458, 2015.

[9] X. Yao, C. C. Huang, and L. Sun, "Two-stream federated learning: Reduce the communication costs," *2018 IEEE Visual Communications and Image Processing (VCIP)*, pp. 1–4, 2018.

[10] Z. Qiao, X. Yu, J. Zhang, and K. B. Letaief, "Communication-efficient federated learning with dual-side low-rank compression," *ArXiv*, vol. abs/2104.12416, 2021.

[11] J. Konecný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *ArXiv*, vol. abs/1610.05492, 2016.

[12] S. Caldas, J. Konecný, H. B. McMahan, and A. S. Talwalkar, "Expanding the reach of federated learning by reducing client resource requirements," *ArXiv*, vol. abs/1812.07210, 2018.

[13] Z. Qiao, X. Yu, J. Zhang, and K. B. Letaief, "Communication-efficient federated learning with dual-side low-rank compression," *ArXiv*, vol. abs/2104.12416, 2021.

[14] G. Yang, K. Mu, C. Song, Z. Yang, and T. Gong, "Ringfed: Reducing communication costs in federated learning on non-iid data," *ArXiv*, vol. abs/2107.08873, 2021.

[15] E. Diao, J. Ding, and V. Tarokh, "Heterofl: Computation and communication efficient federated learning for heterogeneous clients," *ArXiv*, vol. abs/2010.01264, 2021.

[16] Y. Jiang, S. Wang, B. Ko, W.-H. Lee, and L. Tassiulas, "Model pruning enables efficient federated learning on edge devices," *ArXiv*, vol. abs/1909.12326, 2019.

[17] Y. Le Cun, J. S. Denker, and S. A. Solla, "Optimal brain damage," *Proceedings of the 2nd International Conference on Neural Information Processing Systems*, pp. 598–605, 1989.

[18] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, pp. 1135–1143, 2015.

[19] S. A. Aketi, S. Roy, A. Raghunathan, and K. Roy, "Gradual channel pruning while training using feature relevance scores for convolutional neural networks," *IEEE Access*, vol. 8, pp. 171 924–171 932, 2020.

[20] W. Samek, G. Montavon, A. Binder, S. Lapuschkin, and K.-R. Müller, "Interpreting the predictions of complex ml models by layer-wise relevance propagation," *ArXiv*, vol. abs/1611.08191, 2016.

[21] S. Ruder, "An overview of gradient descent optimization algorithms," *ArXiv*, vol. abs/1609.04747, 2016.

[22] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *NeurIPS*, 2019.

[23] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Processing Magazine*, vol. 29, pp. 141–142, 2012.

[24] A. Krizhevsky, V. Nair, and G. Hinton, "The cifar-10 dataset," *online: http://www. cs. toronto. edu/kriz/cifar. html*, vol. 55, no. 5, 2014.