Deep Equilibrium Models Meet Federated Learning

Alexandros Gkillas[1], Dimitris Ampeliotis[2], Kostas Berberidis[1]
[1]Dept. of Computer Engineering and Informatics, University of Patras, Greece
[2]Dept. of Digital Media and Communication, Ionian University, Greece

***Abstract*—In this study the problem of Federated Learning (FL) is explored under a new perspective by utilizing the Deep Equilibrium (DEQ) models instead of conventional deep learning networks. We claim that incorporating DEQ models into the federated learning framework naturally addresses several open problems in FL, such as the communication overhead due to the sharing large models and the ability to incorporate heterogeneous edge devices with significantly different computation capabilities. Additionally, a weighted average fusion rule is proposed at the server-side of the FL framework to account for the different qualities of models from heterogeneous edge devices. To the best of our knowledge, this study is the first to establish a connection between DEQ models and federated learning, contributing to the development of an efficient and effective FL framework. Finally, promising initial experimental results are presented, demonstrating the potential of this approach in addressing challenges of FL.**

***Index Terms*—deep equilibrium models, federated learning, fixed point computation, communication efficiency, implicit layers**

I. Introduction

Federated learning (FL) has emerged as a promising approach for privacy-preserving deep learning by distributing both data collection and model training to the edge. In FL, a group of edge devices, e.g., IoT devices, collaboratively optimize deep learning models without sharing any information about their data. Instead of sending their data, the clients train their models locally and periodically send model updates to a central server for aggregation [1]. In real-world applications FL encounters two significant challenges, namely, the communication burden between the server and the edge devices and the heterogeneity of the devices in terms of computational and power resources [2]–[4]. Recent studies in federated learning have disregarded these two significant limitations [5], utilizing large homogeneous deep neural networks. However, in practical IoT settings, the devices are characterized by limited computational and communication resources, which force them to train smaller and computationally lighter neural networks, thus affecting heavily their performance [4].

To increase communication efficiency in federated learning, compression schemes, e.g., sparsification [6], quantization [7] and client selection [8], have been widely explored. However, these approaches may result in loss of accuracy and introduce bias towards certain devices. Also, focusing on the hardware heterogeneity of the devices, one straightforward approach [9] is to select only clients with adequate computational resources, while disregarding those with limited hardware, which may still possess valuable information. Alternatively, a model architecture could be employed to fit the minimum capabilities of all clients, but this may constrain the overall representation ability of the global model [4]. Another direction relies on deploying different models across clients adapted to their computational resources. To exchange information over heterogeneous models the knowledge distillation technique is applied to enhance the global model with an ensemble of local predictions [9]. However, implementing such approaches can be challenging due to the complex aggregation rules required on the server or the need for clients to share a public proxy dataset, which may not be feasible for devices with limited memory [10], [11].

This work was supported in part by the University of Patras and the RPF, Cyprus, under the project INFRASTRUCTURES/1216/0017 (IRIDA).

Contribution: Unlike the existing literature, in this study, we examine the FL under a different perspective, focusing on the structure and properties of the model employed by the edge devices. To be more precise, rather than employing a conventional deep learning network as referenced in prior work [3], we utilize the Deep Equilibrium (DEQ) models [12]. We argue that these models are characterized by unique properties providing solutions to open-problems in FL including the communication burden between the devices and server and the computational heterogeneity of the local devices.

Although, the Deep equilibrium models have been explored in numerous centralized settings [13]–[17], to the best of our knowledge, our study is the first to investigate a connection between DEQ models and federated learning. More specifically, it is shown here that expressing the entire architecture of a deep learning model as an equilibrium (fixed-point) computation of a single layer or unit (e.g., a residual block) results in an efficient infinite-depth neural network that can offer substantial benefits to federated learning. This compressed representation requires notably less memory, thus enabling efficient communication of model updates between the server and the devices, while achieving competitive performance. Furthermore, the complexity of the DEQ model can adapt dynamically based on the edge devices' computational capabilities by adjusting the number of fixed point iterations required to compute the corresponding equilibrium point. Finally, we propose a novel, weighted average fusion rule that takes into account heterogeneous edge devices that employ different numbers of fixed point iterations, thereby utilizing this information effectively. Note that our proposed method is applicable to any federated learning algorithm.

II. Preliminaries- Deep equilibrium (DEQ) models

The DEQ model is inspired by the observation that a typical k-layer neural network (NN) (with an input-skip connection) [13] can be formulated as follows

$$z_{i+1} = \sigma(\mathbf{W}_i z_i + U_i x + b_i), \quad i = 1, \cdot, k-1 \tag{1}$$

where $\mathbf{W}_i, \mathbf{U}_i$ denote the weight matrices, b_i is the bias term of the $i-$th layer, $\sigma()$ corresponds to the activation function, and x is the input. Under the weight-tying practice [12], $\mathbf{W}_i, \mathbf{U}_i$ and b_i of each layer can be replaced by the same $\mathbf{W}, \mathbf{U}$ and b, thus deriving a weight-tied deep network as follows

$$z_{i+1} = \sigma(\mathbf{W} z_i + \mathbf{U} x + b), \quad i = 1, .., k-1 \tag{2}$$

Note that typically a separate weight $\mathbf{W}_k$ and bias term b_k is employed to generate the final output, since the output can be a different size than the hidden unit. Focusing on the key iteration in (2), the study in [12] recognised that if we apply the transformation (2) for an infinite number of times the output of this transformation should be a fixed point, i.e.,

$$z^\star = \sigma(\mathbf{W} z^\star + \mathbf{U} x + b). \tag{3}$$

In other words, the DEQ models aim to estimate the fixed point $z^\star$ where any further application of transformation (2) would not alter its value. *Note that the above solution corresponds to an* ***infinite-depth*** *network.* In light of this, a DEQ model is defined as a fixed point function $f_\theta(z, x) = \sigma(\mathbf{W} z + \mathbf{U} x + b)$, which obeys the following relation

$$z^\star = f_\theta(z^\star, x). \tag{4}$$

where θ denotes the parameters of the model.

III. Proposed Method

A. Federated Deep Equilibrium Learning

To mathematically formulate the considered federated deep equilibrium learning problem, we define a set of $\mathcal{N}$ edge devices, where each device $n \in \mathcal{N} = \{1, 2, \ldots, N\}$ contains a local private dataset, denoted as $\mathcal{D}_n = \{x_{i,n}, y_{i,n}\}_{i=1}^{p_n}$, where x_n^i is the input, and y_n^i is the corresponding label. Given $\mathcal{D}_n$, each device n aims to train a local DEQ model, whose weights are denoted as θ_n. This can be achieved by minimizing a local objective $g_n(\theta; \mathcal{D}_n)$ that utilizes some loss function, denoted as $\mathcal{L}(.)$. Since, an infinite depth neural network is considered, this minimization process requires the solution of several fixed point problems, needed for the calculation of the output of the DEQ model. In particular, the local objective of device n is

$$g_n(\theta_n; \mathcal{D}_n) = \frac{1}{p_n} \sum_{i=1}^{p_n} \mathcal{L}(z_i^\star, y_{i,n}) \ , \tag{5}$$

where $z_i^\star$ is the solution of the fixed point equation $z_i^\star = f_\theta(z_i^\star, x_{i,n})$, defined in relation (4) and denotes the output of the DEQ model when the input x_n^i is applied. Thus, the goal of edge device n is to obtain the local DEQ model θ_n that minimizes the objective in (5).

Under the FL framework, the devices aim to collaboratively train a global DEQ model, say θ_g, in a manner orchestrated by a central server. Particularly, the FL minimizes the aggregation of the local objectives and entails a common output for all devices using the global model. The objective of FL is

$$G(\theta_g) = \sum_{n=1}^{N} w_n g_n(\theta_n; \mathcal{D}_n) \tag{6}$$

where w_n denote some weight coefficients.

B. Edge device side-local update: forward and backward pass

The core component of the DEQ model is the transformation $f_\theta(\cdot)$ in (3) that is driven to equilibrium. Given this transformation, each device faces the following two challenges during its training/inference procedures. **First**, given an input $x_{i,n}$ a fixed point of $f_\theta(\cdot, x_{i,n})$ is required to be estimated efficiently during the *forward pass*. **Second**, given its dataset $\mathcal{D}_n$, each edge device needs to minimize its local objective (5) to efficiently update the weights of the local DEQ model. Without loss of generality, to simplify the notations and calculations below, we focus only on the edge device n using only a single pair of training samples denoted as $\{x, y\}$.

1) Forward pass - Calculating fixed points: During the training and testing procedures of edge device n, a large number of fixed point iterations needs to be computed based on the transformation map of DEQ model in (4) to estimate the fixed point $z^\star = f_\theta(z^\star, x)$.

A simple yet time-consuming approach to calculate the fixed point is to apply the following recursive scheme, i.e., $z^{(k+1)} = f_\theta(z^{(k)}, x)$ until $z^{(k+1)}$ and $z^{(k)}$ are sufficiently close to each other. In view of this, we employ the Anderson acceleration approach [18], thus accelerating significantly the calculation of the fixed-point. Specifically, the Anderson acceleration method employs m previous fixed point estimates to compute the next fixed point estimate, thus forming the following rule

$$z^{(k+1)} = (1-\beta) \sum_{i=0}^{m-1} \alpha_i z^{(k-i)} + \beta \sum_{i=0}^{m-1} \alpha_i f_\theta(z^{(k-i)}, x), \tag{7}$$

for $\beta > 0$. The vector $\alpha \in \mathbb{R}^m$ is computed via

$$\arg\min_{\alpha} \|\mathbf{Q}\alpha\|_2^2 \, , \quad s.t. \quad 1^T \alpha = 1 \tag{8}$$

where $\mathbf{Q} = [f_\theta(z^k, x) - z^{(k)}, \ldots, f_\theta(z^{k-m+1}, x) - z^{(k-m+1)}]$ is a matrix, which contains m past residuals.

2) Backward pass - Calculating the Gradient: The second challenge that devices encounter during their training process is the computation of backpropagation. The objective is to train their local DEQ model without the need to backpropagate through a significant number of fixed-point iterations.

Let $z^\star = f_\theta(z^\star, x)$ be the fixed point estimated during the forward pass given the input x from the local dataset of device n and $l(z^\star) = \mathcal{L}(z^\star, y)$ be a loss function using only an example-target y. The gradient with respect to the local DEQ model, denoted as θ is

$$\frac{\partial l}{\partial \theta} = \left(\frac{\partial z^{\star T}}{\partial \theta}\right) \left(\frac{\partial l}{\partial z^\star}\right), \tag{9}$$

where the first factor is the Jacobian of $z^\star$ w.r.t. θ and the second factor is the gradient of the loss function. To avoid backpropagating through a large number of fixed point iterations, the implicit backpropagation [12] is performed to estimate the Jacobian of $z^\star$ w.r.t. θ. To this end, we implicitly differentiate both parts of fixed point equation (4). i.e., $z^\star = f_\theta(z^\star, x)$ and we solve for $\frac{\partial z^\star}{\partial \theta}$, thus deriving an explicit expression for the Jacobian

$$\frac{\partial z^\star}{\partial \theta} = \left(I - \frac{f_\theta(z^\star, y)}{\partial z^\star}\right)^{-1} \frac{f_\theta(z^\star, y)}{\partial \theta} \tag{10}$$

Using relation (10), equation (9) is reformulated as follows

$$\frac{\partial l}{\partial \theta} = \frac{f_\theta(z^\star, x)^T}{\partial \theta} \left(I - \frac{f_\theta(z^\star, x)}{\partial z^\star}\right)^{-T} \frac{\partial l}{\partial z^\star}. \tag{11}$$

Thus, we need to compute only a computationally efficient Jacobian-vector product, as shown in equation (11) in order to obtain the gradient of the loss in (9). Following [17], [19], to calculate this Jacobian-vector product, the vector γ is defined

$$\gamma = \left(I - \frac{f_\theta(z^\star, y)}{\partial z^\star}\right)^{-T} \frac{\partial l}{\partial z^\star} \Rightarrow \gamma = \left(\frac{f_\theta(g^\star, y)}{\partial z^\star}\right)^T \gamma + \frac{\partial l}{\partial z^\star}. \tag{12}$$

Note that expression (12) also defines a fixed point equation. Hence, solving this fixed point equation and computing the fixed point $\gamma^\star$, the gradient in (11) can be written as

$$\frac{\partial l}{\partial \theta} = \frac{f_\theta(z^\star, x)^T}{\partial \theta} \gamma^\star. \tag{13}$$

A great advantage is that the Jacobian-vector product in (12) can be efficiently computed by conventional automatic differentiation tools with constant memory [12].

C. Advantages (Properties) of the DEQ models in FL

Communication efficiency: One of the most important benefits that stems from the adoption of a DEQ neural network model in the FL framework, is the significant reduction in the number of model parameters that need to be transmitted. The DEQ model utilized by edge devices can be interpreted as a compressed version of a K-layer deep network in the sense that it only employs a single layer as its central component. This layer defines the transformation function $f_\theta(\cdot)$ that drives the system towards equilibrium. In other words, if we consider that the transformation function $f_\theta(\cdot)$ is defined in terms of M parameters, then the total number of parameters required for the transmission of a DEQ model is somewhat more than M, if we also consider the parameters of the output layer. On the other hand, if a K-layer deep neural network model was employed in the federated learning framework, then each layer typically employs a number of parameters of the order of M per layer and the total number of parameters required for the transmission of one model would be close to $K \cdot M$, i.e., a multiple of what is required by adopting a DEQ model.

Reduced memory requirements: An obvious benefit that results from the adoption of DEQ neural network models in the federated learning framework, is that both the clients as well as the parameter server need a significantly smaller amount of memory, to store and process the models. Thus, this enables the use of devices with smaller costs. If we take into account that some models have several millions (or even billions) of parameters, we realize the importance of such a memory saving.

Support for heterogeneous devices: Another important advantage of the use of DEQ models in the FL framework is that it naturally enables the incorporation of edge devices with significantly different processing capabilities. In order to explain this property, it is important to note that a DEQ model derived by following the procedure in Section III-B corresponds to an infinite depth neural network, if the algorithm used to compute the fixed point performs enough iterations to achieve convergence. Likewise, when the iterative algorithm employed by the devices to compute the fixed point of the transformation $f_\theta(\cdot)$ (see eq. 4) performs a limited number of iterations, this results into a model that approximates the ideal infinite depth neural network. Thus, different devices can perform different numbers of iterations, according to their computational capabilities. Devices with more processing power can perform iterations until convergence is reached. However, less powerful clients can perform fewer iterations, at the cost of reaching approximate, but valid, models. In any case, the DEQ models derived by all heterogeneous devices can be aggregated easily at the server side, since they are all defined in terms of the same transformation $f_\theta(\cdot)$.

D. Server-side - fusion rule

On the server-side, considering that the DEQ models sent by the devices have been derived by heterogeneous edge devices that each may employ a different number of fixed point iterations, a weighted average fusion rule is proposed that effectively utilizes this information. In more detail, at every communication round t, the server employs the following fusion rule, defined by

$$\theta_g^{(t)} = \frac{1}{\sum_{n=1}^{N} k_n} \sum_{n=1}^{N} k_n \theta_n^{(t)} , \tag{14}$$

where $\theta_g^{(t)}$ denotes the global model, $\theta_n^{(t)}$ is the DEQ model sent to the server by client n and k_n denotes the number of fixed point iterations performed by client n.

IV. Experimental part

In this section, we evaluate the effectiveness of the proposed Federated Deep equilibrium learning approach and compare it with the corresponding federated learning approaches using traditional deep learning models in different scenarios. The evaluation is conducted from two perspectives: 1) Communication efficiency and accuracy, and 2) Heterogeneity of edge devices in terms of their computational capabilities.

A. Implementation details

Dataset and settings: We utilize the well-known CIFAR10, adopting similar training/testing splits as previous studies e.g., [20] and exploring two scenarios, i.e, 1) an IID data case and 2) a non IID data case. In the non IID data setting, the label ratios follow the Dirichlet distribution, in which the Dirichlet parameter was set to 0.25 [20]. Since, the focus of this study is to connect the DEQ models with the federated learning, for

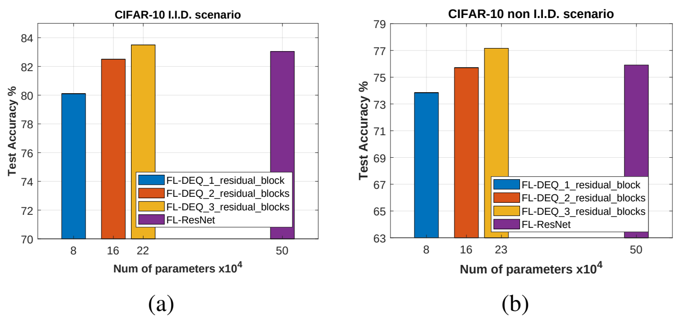


Fig. 1: The accuracy of the proposed Federated Deep Equilibrium Learning approach, which utilizes DEQ models with several residual blocks, is compared to Federated Learning (FL) that employs the ResNet model, for both I.I.D. and non-I.I.D. scenarios.

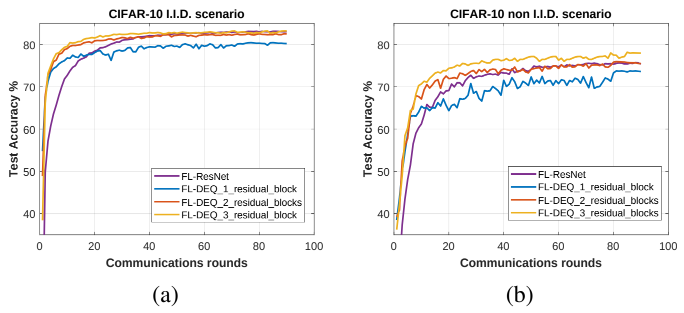


Fig. 2: The convergence of the proposed Federated Deep Equilibrium Learning approach, which utilizes DEQ models with several residual blocks, is compared to Federated Learning (FL) that employs the ResNet model, for both I.I.D. and non-I.I.D. scenarios.

all experiments we employed the FedAvg algorithm with a network of $N = 15$ devices.

Compared deep learning networks: Regarding the compared methods, we consider two federated learning methods based on the FedAvg algorithm that employ two deep learning models, i.e., 1) a ResNet network and 2) a lighter CNN model. The ResNet model comprises five residual blocks and a fully connected (FC) layer with ten neurons as outputs for ten classes. The CNN model consists of L convolutional layers followed by a FC layer, where the value of the parameter L varies from 4 to 10 throughout the experiments.

The choice of DEQ model (transformation $f_\theta(.)$): The core component of the DEQ models is the transformation $f_\theta(.)$. In this study, we explored two DEQ models comprising transformations based on the previously mentioned ResNet and CNN networks. In the first case, since the ResNet consists of residual blocks, we considered a residual block as a transformation f_θ:

$$f_\theta(z, x) = \mathcal{B}(\sigma(z + \mathcal{B}(x + \mathbf{W}_2 * \mathcal{B}(\sigma(\mathbf{W}_1 * z)))))$$

where $\mathcal{B}$ represents the Group normalization and σ is the ReLU. Note that the DEQ model may comprise of additional residual blocks to achieve a balance between accuracy and number of parameters. In the second case, the transformation f_θ of the DEQ model comprises of a convolutional layer (or more). After finding the equilibrium point, the output of the above DEQ models is followed by a FC layer.

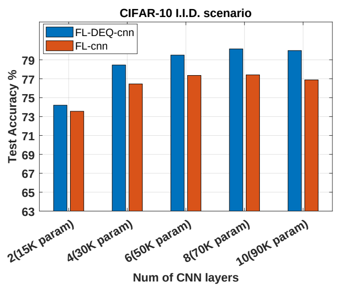
Parameters: Each federated learning method utilized 90 communication rounds between the server and the edge devices. Concerning the local training process of the edge devices, we employed 5 epochs with batch size equal to 32 using the SGD as optimizer. During the training/testing process we used 10 fixed point iterations for the forward and backward passes.

B. Results - Communication efficiency

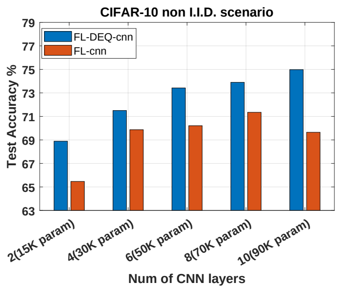
Regarding the ResNet architecture, in Figure 1, we compared the FL-ResNet method with the proposed Federated Deep Equilibrium Learning approach, which utilizes a DEQ model with varying numbers of residual blocks (one, two, or three) denoted as FL-DEQ-1-residual-block, FL-DEQ-2-residual-blocks, and FL-DEQ-3-residual-blocks, respectively. The results validate that the connection of DEQ models with federated learning offers several advantages. In more detail, utilizing only a single residual block in transformation f_θ i.e., the method FL-DEQ-1-residual-block, we are able to achieve more than **84% reduction in the parameters** of the local models with negligible performance loss (less than **3.16%**) for both the I.I.D. and non I.I.D. scenarios as compared to the FL-ResNet approach. Interestingly, with a slight increase in the complexity of the DEQ model (using 3 residual blocks), the FL-DEQ-3-residual-blocks outperforms the FL-ResNet method, while requiring **56% less parameters**. Similar results were obtained for the CNN architecture. Specifically, in Figure 3, the Federated Deep Learning method that uses a transformation function with 4 convolutional layers provides a significant reduction in the number of parameters (**more than 66.5%**) while achieving better performance than the all FL-CNN cases. Additionally, from Figure 2, the three considered federated deep equilibrium learning methods demonstrate faster convergence rates requiring less communication rounds to achieve a satisfactory performance accuracy compared to the FL-ResNet approach. Overall, the proposed FL approach provides significant communication gains in terms of both the number of parameters that need to be exchanged between devices and the server, and the number of communication rounds required.

C. Results - Heterogeneous Devices

In this experiment, we considered a scenario in which 40% of the participating edge devices were characterized by limited computational resources, and therefore performed only 3 fixed point iterations during their local training update. The remaining devices, with more powerful hardware, performed 10 fixed point iterations. We evaluated the performance of the FL-DEQ-2-residual-blocks method in two cases: (1) where all edge devices performed 10 fixed point iterations (homogeneous scenario), and (2) where the number of fixed point iterations varied depending on the computational resources



(a)



(b)

Fig. 3: The proposed Federated Deep Equilibrium Learning approach, which utilizes DEQ models with 2 to 10 convolutional layers, has been evaluated for its accuracy in comparison to Federated Learning (FL) that employs CNN models with several CNN layers, for both I.I.D. and non-I.I.D. scenarios.

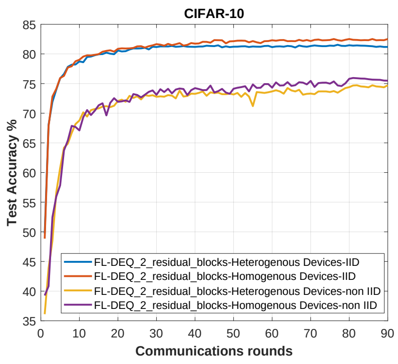


Fig. 4: The FL-DEQ-2-residual-blocks method is evaluated in a heterogeneous scenario, demonstrating resilience to varying computational resources of edge devices without sacrificing convergence speed or accuracy compared to the homogeneous scenario.

of the edge devices (heterogeneous scenario). The results, shown in Figure (4), demonstrate that the proposed approach is resilient to heterogeneity in the hardware capabilities of the edge devices. Specifically, the FL-DEQ-2-residual-blocks method achieved similar accuracy in both the homogeneous and heterogeneous scenarios, despite the fact that some devices performed fewer fixed point iterations than others. This suggests that the proposed approach can effectively leverage the varying computational resources of edge devices, without sacrificing accuracy or convergence speed. Note that similar results were obtained by utilizing the other proposed schemes but they have been omitted due to space limitations.

V. Conclusions

This study presents a novel approach that employs Deep Equilibrium models to address several challenges in federated learning. In particular, it was shown that using DEQ models in the context of FL can effectively handle the communication overhead associated with sharing large models and the computational heterogeneity of edge devices. Promising initial experimental results were presented, indicating the potential of this approach in addressing the challenges of FL. Further experimental work is needed to fully explore the capabilities and limitations of the proposed approach, including its performance on different datasets and neural network architectures.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Artificial intelligence and statistics*. PMLR, 2017.

[2] P. Kairouz, H. B. McMahan, and et.al, “Advances and open problems in federated learning,” *CoRR*, vol. abs/1912.04977, 2019. [Online]. Available: http://arxiv.org/abs/1912.04977

[3] T. Gafni, N. Shlezinger, K. Cohen, Y. C. Eldar, and H. V. Poor, “Federated learning: A signal processing perspective,” *IEEE Signal Processing Magazine*, vol. 39, no. 3, pp. 14–41, 2022.

[4] C. Xu, Y. Qu, Y. Xiang, and L. Gao, “Asynchronous federated learning on heterogeneous devices: A survey,” *CoRR*, vol. abs/2109.04269, 2021. [Online]. Available: https://arxiv.org/abs/2109.04269

[5] J. Wang, Z. Charles, and et.al, “A field guide to federated optimization,” 2021. [Online]. Available: https://arxiv.org/abs/2107.06917

[6] D. Rothchild, A. Panda, E. Ullah, N. Ivkin, I. Stoica, V. Braverman, J. Gonzalez, and R. Arora, “FetchSGD: Communication-efficient federated learning with sketching,” in *Proceedings, Int. Conf. on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 8253–8265.

[7] N. Shlezinger, M. Chen, Y. C. Eldar, H. V. Poor, and S. Cui, “Federated learning with quantization constraints,” in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 8851–8855.

[8] J. Xu and H. Wang, “Client selection and bandwidth allocation in wireless federated learning networks: A long-term perspective,” *IEEE Trans. on Wireless Communications*, vol. 20, no. 2, pp. 1188–1200, 2021.

[9] K. Bonawitz and et. al, “Towards federated learning at scale: System design,” in *Proceedings of Machine Learning and Systems*, A. Talwalkar, V. Smith, and M. Zaharia, Eds., vol. 1, 2019, pp. 374–388.

[10] J. Zhang, S. Guo, X. Ma, H. Wang, W. Xu, and F. Wu, “Parameterized knowledge transfer for personalized federated learning,” in *Advances in Neural Information Processing Systems*, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021.

[11] Z. Zhu, J. Hong, and J. Zhou, “Data-free knowledge distillation for heterogeneous federated learning,” in *Proceedings, Int. Conf. on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 12 878–12 889.

[12] S. Bai, J. Z. Kolter, and V. Koltun, “Deep Equilibrium Models,” in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019.

[13] X. Xie, Q. Wang, Z. Ling, X. Li, G. Liu, and Z. Lin, “Optimization induced equilibrium networks: An explicit optimization perspective for understanding equilibrium models,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 45, no. 3, pp. 3604–3616, 2023.

[14] T. Chen, J. B. Lasserre, V. Magron, and E. Pauwels, “Semialgebraic representation of monotone deep equilibrium models and applications to certification,” in *Advances in Neural Information Processing Systems*, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021.

[15] A. Gkillas, D. Ampeliotis, and K. Berberidis, “A highly interpretable deep equilibrium network for hyperspectral image deconvolution,” in *ICASSP 2023 - 2023 IEEE Inter. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2023, pp. 1–5.

[16] S. Bai, V. Koltun, and J. Z. Kolter, “Multiscale Deep Equilibrium Models,” in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 5238–5250.

[17] A. Gkillas, D. Ampeliotis, and K. Berberidis, “Connections between deep equilibrium and sparse representation models with application to hyperspectral image denoising,” *IEEE Trans. on Image Processing*, pp. 1–1, 2023.

[18] H. F. Walker and P. Ni, “Anderson acceleration for fixed-point iterations,” *SIAM J. on Numerical Analysis*, vol. 49, no. 4, pp. 1715–1735, 2011.

[19] “Deep Implicit Layers - Neural ODEs, Deep Equilibirum Models, and Beyond.” [Online]. Available: http://implicit-layers-tutorial.org/

[20] H.-Y. Chen and W.-L. Chao, “On bridging generic and personalized federated learning for image classification,” in *Int. Conf. on Learning Representations*, 2022.