"Survey research paper on Federated Learning"

Amrita Aash
Dept. of Computer Science
IIIT Delhi
amrita22011@iiitd.ac.in

Shubham Agrawal
Dept. of Computer Science
IIIT Delhi
shubham22124@iiitd.ac.in

I. Introduction

Federated Learning is a distributed machine learning framework wherein various clients collaborate to construct a global model without exposing their private data to the server. In traditional machine learning, data is collected from multiple devices and stored in a central location, and the model is then trained on this centralised data. In federated learning, the data remains on the devices, and each device trains the model locally using its data. The model updates are then sent to a central server, aggregating them to create a single global model. This is especially useful when the data is sensitive or unavailable for aggregation at a central repository, such as data from our smartphones, medical data from hospitals and many more.

The central server, local devices, and the Federated Learning algorithm are the three essential components of Federated Learning. The central server runs the training process and keeps the global model current. Local devices train the model with local data and send updates to the central server. Finally, the Federated Learning algorithm combines local device information and updates the global model. Federated Learning provides various benefits, including data privacy preservation, reduced network communication costs, increased scalability, and allowing devices to profit from the data of other devices while keeping their own data secret.

Federated Learning protects consumers' data privacy by storing data on local devices and not allowing data to leave the user's device. Furthermore, network communication costs are reduced because only model updates, rather than actual data, are exchanged between local devices and the central server. Eventually, Federated Learning can scale up to many devices and is particularly useful in scenarios where data is dispersed across multiple locations.

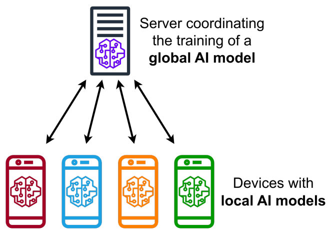

Server coordinating the training of a **global AI model**
Devices with **local AI models**

Fig. 1. Schematic diagram of Federated Learning

II. Literature Review

As Federated Learning deals with data from various sources, heterogeneity and non-IID data are significant issues. Many algorithms have been proposed to aggregate such data in local client updates. FedAvg (FederatedAveraging algorithm) [1] is the most fundamental algorithm on deep networks of federated learning and is based on iterative model averaging. It combines local stochastic gradient descent (SGD) on each client with a server that performs model averaging proportional to the device's training data. The authors performed the experiments in a controlled setting with unbalanced and non-IID data. K clients were chosen, with each having a distinct local dataset. For each round, the server randomly sampled a fraction of C clients to which it communicated the current global model parameters of the algorithm. Each sampled client executes local computation leveraging the global model's state and its own dataset and communicates an update to the server. The server aggregates these updates in the global model and again sends them to each sampled client, and this way, the procedure is iteratively repeated until the global model converges. The above algorithm suffers from client drift, meaning the global model convergence is slow and unstable.

To incorporate this issue SCAFFOLD [2] was introduced. It is another federated learning algorithm which is principally concerned with data heterogeneity. It incorporates control variates (variance reduction) in its local updates to rectify the client drift limitation of FedAvg [1]. The authors proved that SCAFFOLD [2] requires significantly fewer iterations of communication and thus is not much affected by data heterogeneity and client sampling. Intuitively, SCAFFOLD [2] estimates the update direction for the server model as c and the update direction for each client as c_i. It then calculates the difference $(c - c_i)$ as an estimate of the client drift, which is used to rectify the local update.

$$\boldsymbol{y}_i \leftarrow \boldsymbol{y}_i - \eta_l(g_i(\boldsymbol{y}_i) + \boldsymbol{c} - \boldsymbol{c}_i)\,.$$

This process successfully helps to overcome heterogeneity and converges in fewer iteration of communication. To prove the advantage of choosing local steps over large-batch SGD, the authors took the advantage of the similarity between the clients to reduce further rounds of communication. They also showed that, while though SCAFFOLD [2] is usually at least as quick as SGD, it may be considerably quicker depending on the degree of Hessian dissimilarity in the data. Finally, with the help of client sampling and heterogeneous data, SCAFFOLD [2] also derived stricter convergence rates for FEDAvg [1].

Another area for improvement with FedAvg [1] is that it simply discards clients who cannot complete their epochs within a given time, i.e. a participating device cannot do a variable amount of local work based on its underlying system specifications and constraints. To overcome this issue, FedProx [3] was introduced. It is a framework which focuses on system heterogeneity. FedAvg [1] fails to converge in non-IID settings and lacks convergence guarantees. FedProx, a federated optimization algorithm, tackles the above heterogeneity challenges in theoretical and empirical settings. FedProx [3]is defined as a generalization and re-parametrization of FedAvg [1]. It does so by allowing variable amounts of work to be done locally across clients depending on the availability of the resources and then aggregates these partial solutions. Unlike FedAvg [1], FedProx [3] does not discard such devices. The main idea of FedProx [3] is that each of the local objective functions can be solved inexactly, allowing a balance between the local computation done on the client side and the communication between the client and server to adjust, depending on the local iterations. γ-inexactness is used to measure the amount of local computation from each local client in each iteration. γ-inexactness must differ from each device and each iteration as different client devices will make further progress to solve their local subproblem due to system specifications like battery level, device power and many more. The authors also suggested adding a proximal term to the local subproblem to effectively limit the effect of local variable updates. Proximal term is beneficial as it handles the problem of statistical heterogeneity by keeping the local updates and the global initial model closer to each other without worrying about manually setting the local number of epochs. Proximal term also ensures the safe incorporation of local work from varying system heterogeneity. Theoretically, the framework offers convergence assurances when training on data from non-identical distributions (statistical heterogeneity) and when accommodating device-level system limitations by enabling each device to perform varying amounts of work (systems heterogeneity). FedProx [3] showed that it could converge more robustly than FedAvg [1], given a collection of realistic federated datasets. On average, 22% of test accuracy was improved in a highly heterogeneous setting showing a more stable and accurate convergence behaviour than FedAvg [1].

Many efforts have also been made to bring work done in neural architecture search (NAS) to a federated setting 2. This paper introduces a framework, 'FEDNAS' [4], which uses a combination of federated learning and neural architecture search (NAS) to find the best neural architecture to handle the diverse data found on different devices. In a federated setting, the data distribution is concealed from the researchers. They have to devise or select various architectures and adjust hyperparameters remotely to accommodate the dispersed data to attain a superior model architecture with enhanced accuracy. This process is costly because it requires several cycles of training on edge devices, which have far greater connection costs and processing demands than the environment found in data centres. So the authors resorted to neural architecture search in a federated setting. The search space, the searching algorithm, and the performance estimation approach (cite:b5) are the three components that make up NAS. The mixed-operation search space described in DARTS [6] and MiLeNAS [7] is included in their search space. They build it up as an entire model architecture while searching in two shared convolutional cells. They suggested mixed operations utilising softmax across all potential operations to bring the candidate operations between two nodes (such as convolution, max pooling, skip connection, and zero) to a continuous search space. In FEDNAS [4], each device trains a candidate neural architecture on its data and then combines the architectures to create the final global model. They show that FEDNAS [4] performs better than current federated learning methods through experiments on well-known datasets.

In order to find a neural architecture customized for each device, the paper uses an improved version of the gradient-based neural architecture search method called MiLeNAS [7]. This method is well-suited for edge devices that have limited computational resources. After conducting a local search, each device sends the model architecture and weights to the server. The server then combines the weights and architecture using a weighted aggregation method to create the updated server-side model. The updated model is returned to each device for further exploration in the next round. This approach leads to better model performance and more excellent resistance to data heterogeneity. Their experiments were directed towards finding an optimal architecture α and corresponding model parameters w that can fit the non-IID dataset distributed among many workers more efficiently and thus achieve better model performance. They concentrated on looking for CNN architecture to improve the efficiency of the picture categorization process. The paper introduces an AutoFL system that utilizes FedNAS [4] and is built on the FedML research library for federated learning. The AutoFL system consists of two main components: the Server Manager and the Client Manager. These components are encapsulated as ComManager, Trainer, and Aggregator, which provide high-level APIs for the different layers. This system can be used to solve problems in the FedNAS [4] setting. In future works, they suggest merging search and evaluation into one stage. To address the suboptimal performance of Federated Learning with predefined models on non-iid data, [8] developed a method to directly search for global and personalized models using federated NAS. They noted that most of the NAS methods are computationally very expensive and need further caliberations after the search, making the process incresingly complex. Their method, which is a one-stage search method with hardware-awareness and low computational cost, enhances the computational efficiency of NAS in federated setting. For performing NAS, they opted for the parent-child network approach, in which child networks are sampled from a super-network(parent). A directed graph is taken as the main network. Children are sampled using DSNAS [9], which is an end to end NAS method for sampling child network in one stage. The authors proposed RaFL (Resource aware Federated Learning) [10] to address the issues of data and system heterogeneity. The idea was to allocate different neural network architecture to different clients based on their compute capacities and local data distribution using NAS, through knowledge extraction and fusion. All the clients in the RaFL server also get a shared smaller "student" network that acts as a knowledge bridge in the diverse network setting. Through deep mutual learning, RaFL clients co train their network pairs and infuse knowledge into their knowledge network. Local knowledge from clients is aggregated to form global knowledge and is transferred to edge devices. The authors claim that their method allows considerable drop in communication overhead, while taking into account device and data heterogeneity. For NAS, Once-for-All weight sharing super networks were used, from which heterogeneous subnets were sampled and deployed on edge clients.

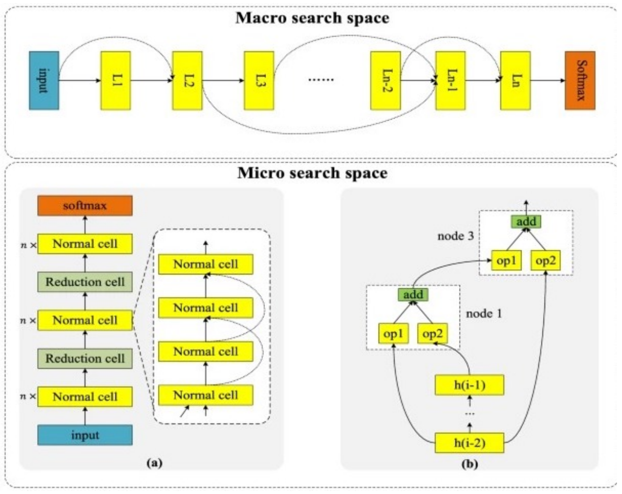


Fig. 2. Federated Learning with Neural architecture search

In a federated learning setup many clients have redundant gradients3. In the paper "Towards Understanding Biased Client Selection in Federated Learning" [11], the authors examined how choosing some clients over others affects how fast the model learns. They also showed mathematically that picking clients with more significant local errors makes the model learn faster. They suggested a way to pick clients based on how well the model is doing globally and used the Power-of-Choice approach, which is a fast and flexible way to pick clients that balance speed and quality. They tested their approach and found that Power-of-Choice can learn up to three times faster and get 10% better accuracy than picking clients randomly. They also made some variations of Power-of-Choice that use less communication and computation resources. Instead of evaluating the loss in the entire local dataset, they used mini-batches to approximate it to save local computation costs. By changing the size d of the candidate client set A, the authors illustrated the trade-off between convergence speed and solution bias. The selection skew increases as d increases, leading to faster error convergence and a higher error floor. If d = m (m = max(CK, 1), K is the total clients, and C is the proportion of clients to be sampled), random sampling without replacement was achieved. The selection skew increases as d increases, leading to faster error convergence and a higher error floor. d = m shows random sampling without replacement in the proportion of pk (proportion of training data in client k). They proved that even with three times fewer clients in each round than random picking, Power-of-Choice could learn two times faster and get 5% better accuracy. The paper's [14] authors suggest a method for federated learning that involves selecting a subset of clients with diverse and representative gradient information. Only the updates from this subset are transmitted to the server, aiming to approximate the results obtained from aggregating all client information. This is achieved by maximizing a sub modular facility location function defined over gradient space. The method is "federated averaging with diverse client selection (DivFL)." Submodularity models diversity through the property of diminishing returns. As more and more elements are added to the subset, the marginal gains that a new element brings to the set decrease, and this is the indicator of diversity. In DivFL, the server samples a subset

$$S \leftarrow S \cup k^*, \quad k^* \in \underset{k\in \mathrm{rand}(V\setminus S,\ \mathrm{size}=s)}{\operatorname{argmax}} [\bar{G}(S) - \bar{G}(\{k\}\cup S)]$$

of K active clients using a stochastic greedy algorithm sends weights w to them. The clients solve the local subproblem inexactly through Stochastic Gradient Descent, updating local mini-batch weights.

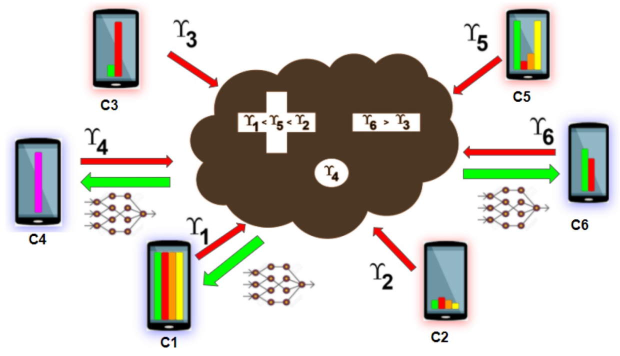


Fig. 3. Various clients and their gradients

Although the data does not leave the client, it does not mean that federated learning is entirely immune to privacy

and security issues. Various types of research have revealed new privacy and security threats, proving that even the model update information can uncover private information about the clients or inject security threats. Malicious servers, clients and even eavesdroppers in the channels may contribute to loss of privacy as shown in figure **??**. The paper [12] demonstrates a significant security flaw in federated learning, wherein an attacker can obtain complete access to private data by analyzing model update information in just a few rounds of iterations. The authors of the study created dummy gradients using fictitious input and label data, which were then used in the standard training process. In a typical federated setup, the focus is on optimizing local weights (model updates); however, the dummy inputs and labels were optimized instead. This resulted in the dummy gradient becoming increasingly similar to the actual gradient, eventually leading to the full exposure of an individual client's original training data. It is possible for model updates to be manipulated during communication between clients or edge devices and the server. Furthermore, the server, which stores the global model and the aggregation of all local updates, is accessible to all users, making it vulnerable to attacks. If an attacker gains access to the server, they could tamper with the global model and send it back to each client, leading to the degradation of the global model as the local weights are no longer accurate. These findings emphasize the importance of implementing robust security measures to safeguard sensitive data in federated learning systems.
By encrypting the parameter exchange between the client and aggregating server, homomorphic encryption is a practical method for enhancing privacy in federated learning. This strategy avoids disclosing model changes by allowing aggregation to be done directly on encrypted parameters. Prohibiting the spoofing or manipulation of model updates improves FL's security. BatchCrypt, an effective homomorphic solution proposed in the paper [13], considers the approach's processing and communication costs. New quantization, encoding, and gradient clipping algorithms were created to employ BatchCrypt. Using BatchCrypt to encrypt an encoded batch of quantized gradients instead of homomorphic encryption on individual gradients resulted in a considerable speedup in training and decreased communication cost with a minor loss in accuracy.

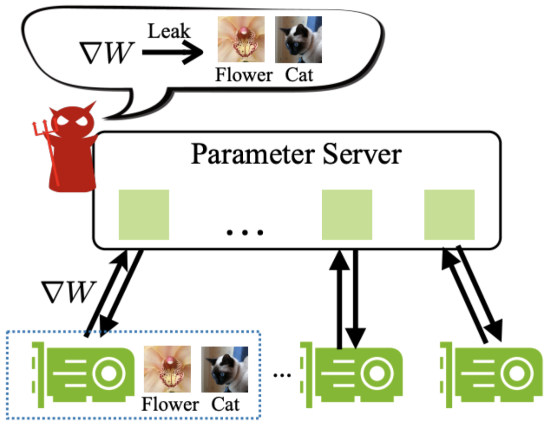


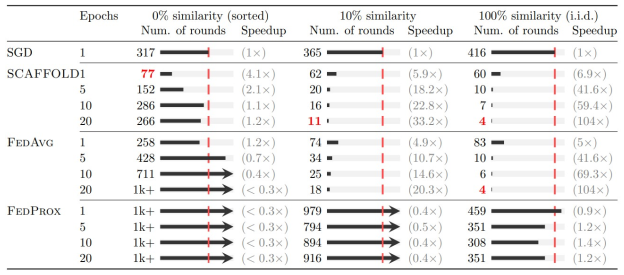
Fig. 4. In the training process attackers are able to get access of the local weights of the clients [12]

III. Methods and Results

In the following section we will discuss the methodology and result of Federated Learning through four different aspects.

A. Federated Learning Algorithms

This research aims to determine the number of communication rounds required to reach 0.5 test accuracy for logistic regression on EMNIST, by varying the number of epochs as seen in the figure 5. The local update methods are based on 1 epoch equivalent to 5 local steps, with a 20% client sample per round. To ensure comparability, we set μ = 1 for FEDPROX and use THE variant two for SCAFFOLD algorithm. According to the study, SCAFFOLD is the most efficient method across all parameters, both in terms of epochs and similarity. With a similarity of 0, which means that data is sorted, FEDAVG underperforms and is slower than SGD when increasing the number of epochs. On the other hand, SCAFFOLD initially performs worse than SGD, but stabilizes and remains faster than it. When similarity increases, meaning the data is more shuffled, both FEDAVG and SCAFFOLD outperform SGD, with SCAFFOLD remaining superior to FEDAVG. Additionally, both approaches gain by extending the number of epochs as similarity rises.



	Epochs	0% similarity (sorted)		10% similarity		100% similarity (i.i.d.)	
		Num. of rounds	Speedup	Num. of rounds	Speedup	Num. of rounds	Speedup
SGD	1	317	(1×)	365	(1×)	416	(1×)
SCAFFOLD	1	77	(4.1×)	62	(5.9×)	60	(6.9×)
	5	152	(2.1×)	20	(18.2×)	10	(41.6×)
	10	286	(1.1×)	16	(22.8×)	7	(59.4×)
	20	266	(1.2×)	11	(33.2×)	4	(104×)
FedAvg	1	258	(1.2×)	74	(4.9×)	83	(5×)
	5	428	(0.7×)	34	(10.7×)	10	(41.6×)
	10	711	(0.4×)	25	(14.6×)	6	(69.3×)
	20	1k+	(< 0.3×)	18	(20.3×)	4	(104×)
FedProx	1	1k+	(< 0.3×)	979	(0.4×)	459	(0.9×)
	5	1k+	(< 0.3×)	794	(0.5×)	351	(1.2×)
	10	1k+	(< 0.3×)	894	(0.4×)	308	(1.4×)
	20	1k+	(< 0.3×)	916	(0.4×)	351	(1.2×)

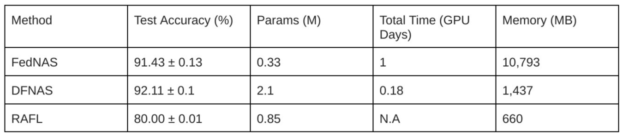
Fig. 5. Comparison of various Federated Learning Algorithms [2]

B. Federated Learning and Neural Architecture Search

The dataset used for experiments tabulated in the above table 2 was CFAR-10. The data distribution was kept non-IID. Clearly, DFNAS produces more accurate architectures, but RAFL is more memory efficient (fewer params and memory).

C. Client Selection in Federated Learning

The performance of DivFL, random sampling, and power-of-choice on synthetic IID data were compared in this study by analyzing their training loss and the mean and variance of test accuracies (referring to figure 7. DivFL utilized only the



Method	Test Accuracy (%)	Params (M)	Total Time (GPU Days)	Memory (MB)
FedNAS	91.43 ± 0.13	0.33	1	10,793
DFNAS	92.11 ± 0.1	2.1	0.18	1,437
RAFL	80.00 ± 0.01	0.85	N.A	660

Fig. 6. Various architectures produced by NAS methods compared in terns of accuracy, parameters, total training time and memory requirements [8]

gradients from clients who participated in the previous round (without overhead). The findings indicate that DivFL outperforms random sampling and power-of-choice, achieving faster convergence and more accurate, slightly more fair solutions.

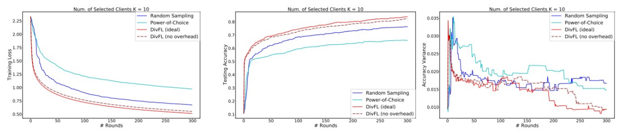


Fig. 7. Training loss vs rounds for Random Sampling, Power-of-Choice and DivFL [14]

D. Privacy and Security in Federated Learning

Privacy and security are the critical features of Federated Learning. Since edge devices send only the updated weights to the server in an encrypted manner, at first glance, FL can think to be free from any malicious attacks as the sensitive data never leaves the actual device. However, figure 8 from [12] shown below shows that we can access the entire data simply through the gradients. [13] presents a new architecture and an algorithm, namely "BatchCrypt", based on homomorphic encryption to reduce such data leaks by malicious attackers. Figure 9 represents the architecture of BatchCrypt.

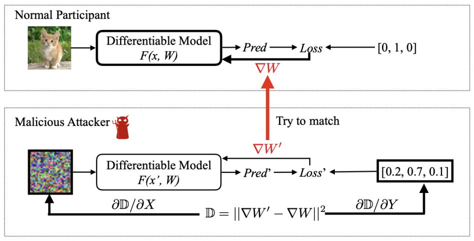


Fig. 8. The figure shows that the attacker is able to get access to the client's sensitive data through the gradient updates only [12]

IV. Discussion and Future Work

Since its genesis in 2017 by Google, federated learning has seen tremendous growth and has been extensively researched. However, a lot is still left to be desired, and the research community is actively working towards making federated learning reliable, accurate and secure. Hot Areas of research in federated learning include global model fusion, communication schemes, asynchronous optimization and granular privacy guarantees. A few of the broad areas in Federated Learning which are still open for discussion are:

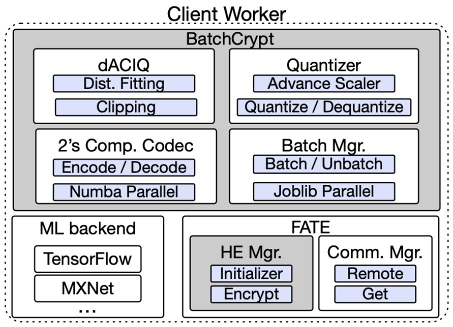


Fig. 9. Architecture of BatchCrypt [13]

1) dealing with dynamic and asynchronous client participation
2) ensuring fairness and accountability
3) incorporating domain knowledge and transfer learning in federated settings
4) evaluating and bench-marking federated learning systems

V. Conclusion

In this paper, we presented some of the latest works on federated learning research. Our discussion was centred around four aspects: FL optimization and aggregation algorithms, client selection, neural architecture search in FL setting and privacy/cyber security concerns of FL. We compared various methods outlying the above aspects, their pros, cons, and practical feasibility. Federated learning has gained attention as a viable data privacy and security paradigm. When users have limited data to train good models, collaborative learning can combine an integrated model and update multiple user models without revealing the source data. Federated learning can also offer a secure method of model sharing and migrating models to specific tasks to address the issue of insufficient data labels. Compared to classical privacy-preserving learning and typical distributed data centre computing, federated learning has distinct advantages and disadvantages. This study provides an overview of federated learning and its applications in various contexts, including the healthcare sector. Recent developments and various open issues and concerns related to federated learning are highlighted, and a list of open issues requiring further investigation is provided. Finding answers to these issues will require the involvement of various research communities.

References

[1] McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In Proceedings of AISTATS, pp. 1273–1282, 2017.

[2] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh. 2020. SCAFFOLD: Stochastic controlled averaging for federated learning. In Proceedings of the International Conference on Machine Learning, Vol. 119. 5132–5143.

[3] Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A. & Smith, V. Federated optimization in heterogeneous networks. arXiv preprint arXiv:1812.06127 (2018).

[4] He, C., Annavaram, M. & Avestimehr, S. Fednas: Federated deep learning via neural architecture search. https://sites.google.com/view/cvpr20-nas/ (2020).

[5] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren, editors. Automatic Machine Learning: Methods, Systems, Challenges. Springer, 2019.

[6] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. arXiv preprint arXiv:1806.09055, 2018.

[7] Chaoyang He, Haishan Ye, Li Shen, and Tong Zhang. Milenas: Efficient neural architecture search via mixed-level reformulation. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[8] Garg, Anubhav, Amit Kumar Saha, and Debo Dutta. ”Direct federated neural architecture search.” arXiv preprint arXiv:2010.06223 (2020).

[9] Hu, S., Xie, S., Zheng, H., Liu, C., Shi, J., Liu, X., and Lin, D. Dsnas: Direct neural architecture search without parameter retraining. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 12081–12089, 2020.

[10] Yu, Sixing, Phuong Nguyen, Waqwoya Abebe, Justin Stanley, Pablo Munoz, and Ali Jannesari. ”Resource-aware heterogeneous federated learning using neural architecture search.” arXiv preprint arXiv:2211.05716 (2022).

[11] Cho, Yae Jee, Jianyu Wang, and Gauri Joshi. ”Towards understanding biased client selection in federated learning.” In International Conference on Artificial Intelligence and Statistics, pp. 10351-10375. PMLR, 2022.

[12] Zhu, Ligeng, Zhijian Liu, and Song Han. ”Deep leakage from gradients.” Advances in neural information processing systems 32 (2019).

[13] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, “BatchCrypt: Efficient homomorphic encryption for cross-silo federated learning,” in Proc. USENIX Annu. Tech. Conf., 2020, pp. 493–506.

[14] Balakrishnan, Ravikumar, et al. ”Diverse client selection for federated learning via submodular maximization.” International Conference on Learning Representations. 2022.