Federated Learning - A Detailed Overview

Type: Survey Paper

Area 5: Federated Learning

Archana Subramanian: archanas@sfu.ca
Harikrishna Karthikeyan: hkarthik@sfu.ca

Abstract

Federated learning is a machine learning technique that allows large-scale applications to be developed by training on data that is local to and distributed across multiple remote nodes, devices and data centres such as mobile phones, IoT connected devices, hospitals, etc. The training process which involves data from massive, heterogeneous data sources leads to unprecedented challenges which makes traditional statistical modelling approaches for large-scale machine learning, optimization and privacy preserving analysis techniques unsuitable. In this survey, we aim to provide an overview of federated learning, the unique characteristics and challenges it poses, and a comprehensive explanation of the different approaches to federated learning along with their respective use-cases. Through this, we aim to achieve a wider adoption of federated learning related techniques in this data driven world.

1 Introduction

Mobile phones, wearable smart devices, and other IoT devices are just a few of the modern distributed networks generating a large amount of data at unprecedented levels. With advancements in technology, the fact that these devices are growing in their computation power coupled with growing concerns over data privacy and transmitting sensitive private[8] information, the world is slowly moving away from central server-based computations to performing them on these edge nodes instead. Federated learning provides a privacy-preserving mechanism to make use of such distributed data and computation resources to train machine learning models. The main idea behind federated learning is to have each edge node learn on its own locally stored data and not have to share the data with anyone else.[12]

Federated Learning allows user devices (edge nodes) to perform most of the computation and a central server updates the model parameters using the gradients returned by the user devices. In spite of FL addressing some of the biggest requirements of the current generation, Federated Learning is not adopted widely. This can be explained by analyzing some of the unique characteristics of Federated Learning that sets it apart from standard parallel optimization techniques.

First, the training data is massively distributed over a very large number of devices and the connection between the central server and the device is not reliable. One of the biggest consequences

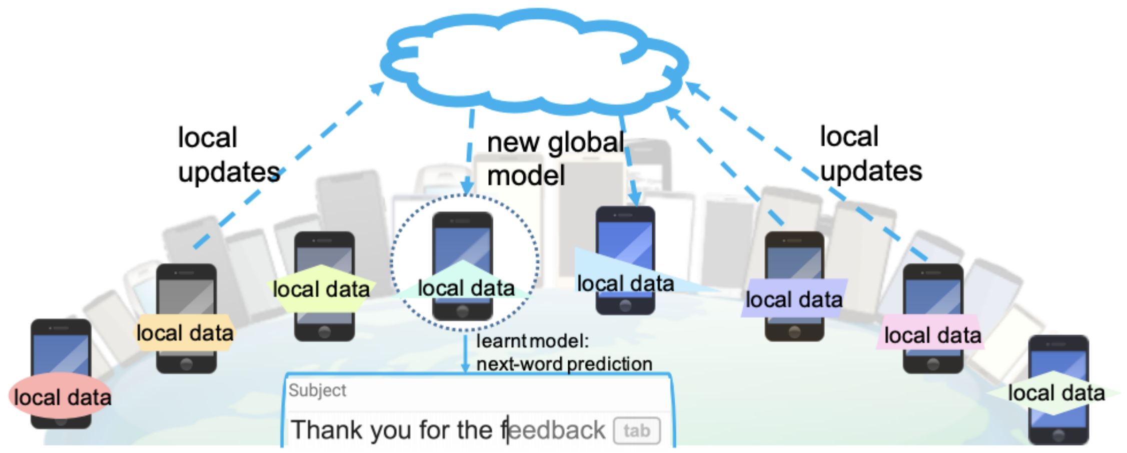


Figure 1: An example of a federated learning task - next word prediction on mobile phones. The above set-up shows that the devices communicate with a central server from time-to-time to learn the global model. At one round, a subset of all available devices perform training on the data available locally. The updates are then sent to the central server. The server then communicates these updates to the remaining devices. This iterative process continues until it encounters a stopping condition or when convergence is attained.[9]

of this is slow communication, which motivated research into communication efficient federated learning. FedAvg is perhaps the first and most widely used federated learning algorithm. This algorithm runs a few steps of SGD in parallel on a small subset of devices and then averages the model updates via a central server periodically. This can lead to more local computation and less communication.

Second, federated learning systems do not have any control over the user devices, unlike traditional distributed learning systems. For example, when the user switches off the phone or WiFi is not available, the connection to the central server is lost. This can lead to the server waiting forever for the local training results to get updated. This can lead to a lot of overhead in the server. The only real option is to wait or ignore the results from the non-responsive devices (called stragglers).

Third, the training data are usually non-IID[4] (Independent and Identically Distributed), that is, the local data on the device are not necessarily samples from the overall distribution. The data available locally depends on the method of collection, storage, etc, and would most likely not represent the overall statistical distribution. This leads to a domain shift[13] between nodes. For example, one user might take photos indoors whereas the other user might take all photos outdoors. This creates problems during theoretical analysis as well as algorithm design. It is important to address the problem of transferring knowledge from the decentralized nodes to a node having a different data domain without the need for human supervision.

Paper Organization: We talk about related work in the field of Federated Learning in **Section**

2. We detail our contributions in the form of this survey in **Section 3**. In **Section 4**, we talk about the method development and detail how our contributions are supported. **Section 5** is reserved for some model evaluations.

2 Related Work

The fundamentals of federated learning as given by Sánchez Clemente et al in their paper "Federated Learning: Challenges, Methods, and Future Directions"[9] discuss what federated learning is - local training of data and communication of results to the central server. It also discusses the unique challenges of federated learning that deals with heterogeneity of data, privacy preservation concerns and communication overhead of the results. However, the actual methods to overcome these problems were brief theoretical explanations.

One of the first methods to overcome the heterogeneity and reduce communication burden was discussed in "FedAvg: Communication-Efficient Learning of Deep Networks from Decentralized Data", the contribution of Ünay et al[12]. The algorithm explored in detail was "FedAvg" that averages the local model parameters element-wise, the weights of which are proportional to the size of the training data (corresponding client data). However, one major drawback of this is that it cannot handle the heterogeneity of the data from various clients and may result in the dropping of those clients that do not adhere to its statistical model expectations. A minor modification of the FedAvg algorithm whereby adding a new proximity parameter to the objective function increases the heterogeneity of the model developed was discussed in "Federated optimization in heterogeneous networks" which also discusses that adding the new term significantly reduces the communication burden.

A slightly better modification of the "FedAvg" algorithm was the "FedMA" algorithm which is briefed in "FedMA: Federated Learning with Matched Averaging"[16]. This is similar to the FedAvg in a way that this method also computes the averages of the models, however, the difference is that it matches the weights on the basis of layers. For instance, the first client data matches with the weights of the first layers and so on. On using this with modern neural network models like CNN and LSTM, it was observed to perform better than other state-of-the-art federated learning techniques and also considerably reduce the communication overhead.

Another method to overcome the challenges of federated learning is "Federated Averaging with Spreadout (FedAwS)" [3]as elaborated in the research paper - "FedAwS: Federated Learning with Only Positive Labels", is used specifically for handling only the positive labels. In this algorithm, the server imposes a geometric regularizer after each round to encourage classes to be spread out in the embedding space. However, one drawback of this algorithm is that the server cannot communicate the complete results to the user. At one time, if a user "i" updates the model, he only has access to the instances from the i-th class along with the class embedding vector W_a.

The other important challenge - the system heterogeneity was discussed in "Federated Multi-Task Learning"[15][2]. In this, the system level variations (hardware differences) were overcome by modelling each client individually using a multi-task learning set-up where multiple similar models are trained together. Similar models are combined based on their related tasks. This when modeled as a group improves the performance. The privacy issues are dealt with in "Secure[17], privacy-preserving and federated machine learning in medical imaging". Given the importance of privacy of medical data[14], this paper explains how data can be masked using one of the many

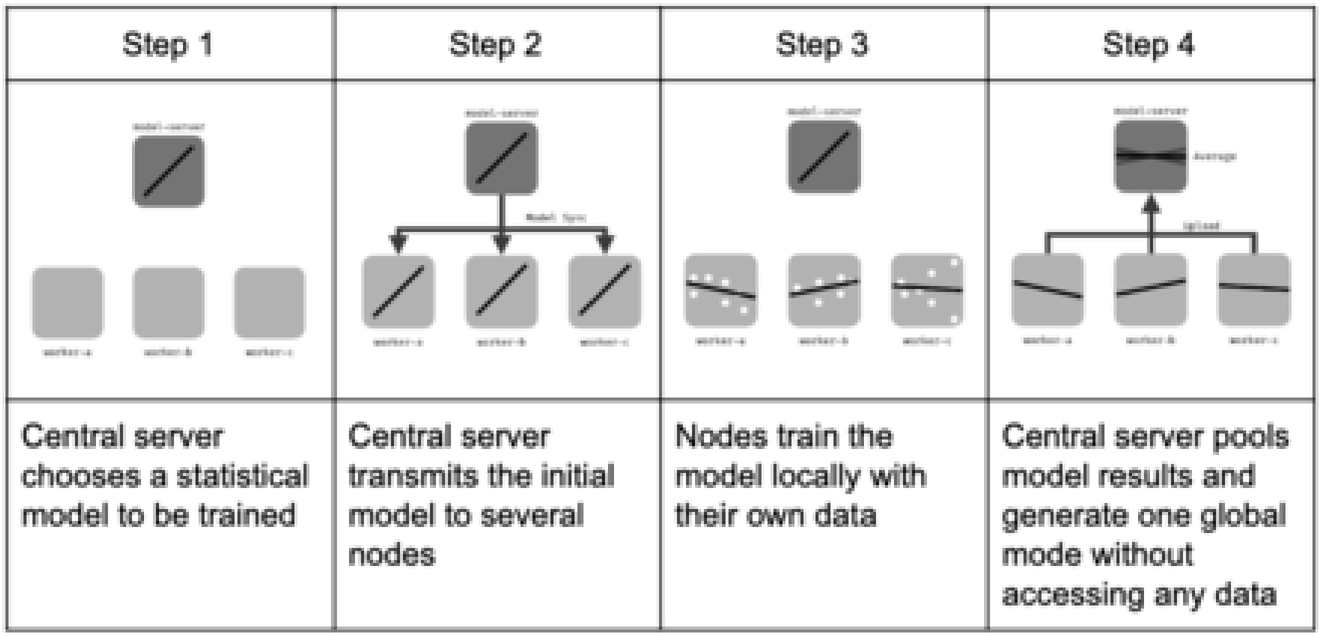

Step 1
Step 2
Step 3
Step 4
model-server
Model Sync
Average
Upload
worker-a
worker-b
worker-c
Central server chooses a statistical model to be trained
Central server transmits the initial model to several nodes
Nodes train the model locally with their own data
Central server pools model results and generate one global mode without accessing any data

Figure 2: A general working of federated learning.[1]

methods suggested - anonymization[6], pseudonymization, encryption, differential privacy, etc.

3 Contribution

Federated learning is a fairly new domain that has gained its popularity over the last couple of years. The resources and research papers available thus far analyse each method of overcoming the shortcomings of federated learning, individually. In this report, we have attempted to compare the most commonly used algorithms. Given that it is a consolidated collection of all important federated learning algorithms, it can be considered as a one-stop solution for any optimization related issues. We elaborate how each method contributes to overcome the shortcomings of other algorithms, namely, heterogeneity, communication burden and privacy preservation, etc.

4 Method Development

4.1 FedAvg

Federated Averages or FedAvg is the most commonly used Federated Learning approach which builds on the earlier FedSGD algorithm. It makes use of Stochastic Gradient Descent in batches from clients to update the central model and redistributes those to the clients for further updates. This suffers from non-IID data as well as other privacy concerns. Other models build on FedAvg and improve it as we will see in the later sections.

4.2 FedMA

FedMA abbreviated as "Federated learning with Matched Averaging" is a modification of the traditional FedAvg algorithm. It is mainly designed for federated learning using modern neural network architectures mainly consisting of CNN (Convolution Neural Nets) and LSTM (Long Short Term Memory). Unlike the FedAvg algorithm which averages all the results of individual training models, the FedMA algorithm constructs the global model in a layer-wise manner. It matches and averages information in the hidden layer on the basis of similar feature extraction principles. This hidden layer information for our modern neural network architectures may include Channels for CNN and hidden states for LSTMs.

Algorithm: Handling of matched averages recursively over a set of devices may lead to poor results. To overcome this challenge, the FedMA algorithm establishes a layer-wise mapping scheme. At first, the first data center gathers the first layer information (of CNN or LSTM). It performs first-layer matching and obtains the weights of the first layer. The data center then broadcasts these weights to all the clients in the network which use these weights to train their data locally. This training happens only on all the consecutive layers, leaving the matched layers frozen. This algorithm is recursively applied until the last layer, where a weighted average based on the class proportions are computed for each client.

Formula:

$$min_{\pi_{li}^{j}} \sum_{i=1}^{L} \sum_{j,l} min_{\theta i} \pi_{li}^{j} c(w_{jl}, \theta_i) s.t \sum_{i} \pi_{li}^{j} = 1 \forall j, l; \sum_{l} pi_{li}^{j} = 1 \forall i, j \tag{1}$$

In the above equation, L is the number of clients, j is the hidden layer information.

Effect on communication: At the beginning of a new round, the global model shares the matched model to all the local clients. The local clients then reconstruct their data to fall in line with the matching results of the global model in the previous round as received. By doing this, the size of the local model and global model are bound to remain in sync and are both kept small. This is better than assigning a large size by utilizing the fully matched global model as the starting point for all client models on every round.

Advantages:
- FedMA outperforms CNN and LSTM
- Significant reduction in communication burdens due to small size updates.

Drawbacks:
- Effect on huge datasets unexplored.

4.3 FedProx

The two main challenges that distinguish federated learning from traditional distributed optimization are high degrees of system and statistical heterogeneity. To tackle this heterogeneity in federated networks both theoretically and empirically, [10] introduces FedProx. FedProx can be viewed as a generalization and re-parameterization of FedAvg, the current state-of-the-art method for federated learning. In the context of systems heterogeneity, FedAvg does not allow participating devices to perform variable amounts of local work based on the constraints of their underlying system; more commonly, it is okay to remove the devices that fail E epochs within a specified time window.

Dropping stragglers (as in FedAvg) or naively incorporating partial information from stragglers (as in FedProx with the proximal term set to 0) implicitly increases statistical heterogeneity and can adversely impact the convergence behavior. To mitigate this issue, they propose adding a proximal term to the objective that helps to improve the stability of the method. This term provides a principled way for the server to account for heterogeneity associated with partial information. Theoretically, these modifications will provide convergence guarantees for this method and to analyze the effect of heterogeneity. Empirically, one demonstrates that the modifications improve the stability and overall accuracy of federated learning in heterogeneous networks—improving the absolute testing accuracy by 22% on a highly heterogeneous setting.

Algorithm: In FedProx, they generalize FedAvg by allowing for variable amounts of work to be performed locally across devices based on their available systems resources and then aggregate the partial solutions sent from the stragglers. In other words, instead of assuming a uniform Gamma for all devices throughout the training process, FedProx implicitly accommodates variable Gammas for different devices and at different iterations. To address the heterogeneity, they propose to add a proximal term to the local subproblem to effectively limit the impact of variable local updates. In particular, instead of just minimizing the local function Fk().

The proximal term is good because it addresses the issue of statistical heterogeneity by restricting the local updates to be closer to the initial (global) model without any need to manually set the number of local epochs. It allows for safely incorporating variable amounts of local work resulting from systems heterogeneity.
Formula:

$$min_w h_k(w; w^t) = F_k(w) + \frac{\mu}{2} - \left\| w - w^t \right\|^2 \tag{2}$$

4.4 FedAwS

Let us consider the scenario of learning a multi class classification model in the federated learning setting where each user only has access to positive data associated with a single class. As a result, during each round of federated learning, they need to update the classifier locally without having access to the features and the model parameters for the negative classes. Conventional distributed algorithms like distributed Stochastic Gradient Descent or federated learning algorithms like Federated Averaging would lead to poor results. For embedding based classifiers, the result is even worse as all the embeddings might collapse onto a single point in trying to minimize the loss function. The author proposes a new algorithm called Federated Averages with Spreadout (FedAwS), where the server imposes a geometric regularizer after each round to spread out the embeddings of the classes.

The important points to notice are that the users are not allowed to communicate with each other, nor do they have access to the classification model parameters associated with other user's classes. This can be explained by the example of Facial recognition models or speaker recognition models which are considered to be sensitive information which cannot be shared.

Unique challenges this poses – the server cannot communicate full model to the user, and when i-th user updates the model he only has access to the set of instances x from the i-th class along with the class embedding vector W_i. Typical loss function does 2 things - increase similarity between instance and positive class embeddings as large as possible, and decrease similarity with negative embeddings. In each communication round, the i-th client has access to

- n_i instance and label pairs with the same label i.
- It's own class embeddings, W_i
- The current instance embedding model parameter θ.

Here without the negative labels, the loss function can only attempt to get the instance embeddings and positive class embeddings as close to each other as possible. In addition to federated averaging, the server in this case performs an additional step to optimize the class embedding matrix such that the class embeddings are separated from each other by a margin v. The spreadout regularizer takes the form:

$$reg_{sp}(W) = \sum_{c \epsilon [C]} \sum_{c' \neq c} (max[0, v - d(w_c, w_{c'})])^2 \tag{3}$$

4.5 SCAFFOLD

In earlier works, Federated Averaging was the default choice of Federated learning algorithm but it does come with its own set of caveats. One of the important reasons FedAvg doesn't perform is if there are any heterogeneous data and this could lead to slower and tight convergence. For our purposes, ideally, the convergence rate should be fast and smoother to avoid 'client-drifting'. This technique SCAFFOLD uses a variance reduction method to auto-correct the potential 'client-drift' whenever the model communicates with other nodes in the network updating their gradient values.

The SCAFFOLD[7] ultimately allows the nodes to build a model that uses the data from other nodes while maintaining privacy by using gradient averages instead of data to create and update the control variable. Each node has its own control variable (variance reduction) which is a prediction of all the data on all the nodes which communicate with the other models in the network. Each node updates its control variable whenever the server updates the client with this information. The control variable essentially holds the gradient of all the nodes. At the starting, it is usually initialized to 0 and updates itself when talking with other nodes in the network. After these updates, the local gradient is self-corrected using the control variable and ensures that the future updated moves towards the true optimum value instead of client optimum value.

By assuming that there are no communication costs penalties, SCAFFOLD can ensure that each update computes an unbiased gradient for heterogeneous data which performs excellently. However, this might not be the ideal situation, and we mimic the ideal update with the help of control variables and making the local updates in sync and converge for heterogeneous clients.

To further back the claims of the effectiveness of this method, the authors conducted experiments on the EMNIST dataset. With the main goal of minimizing the number of communication rounds and getting better results, the experiments prove that SCAFFOLD consistently outperforms SGD,

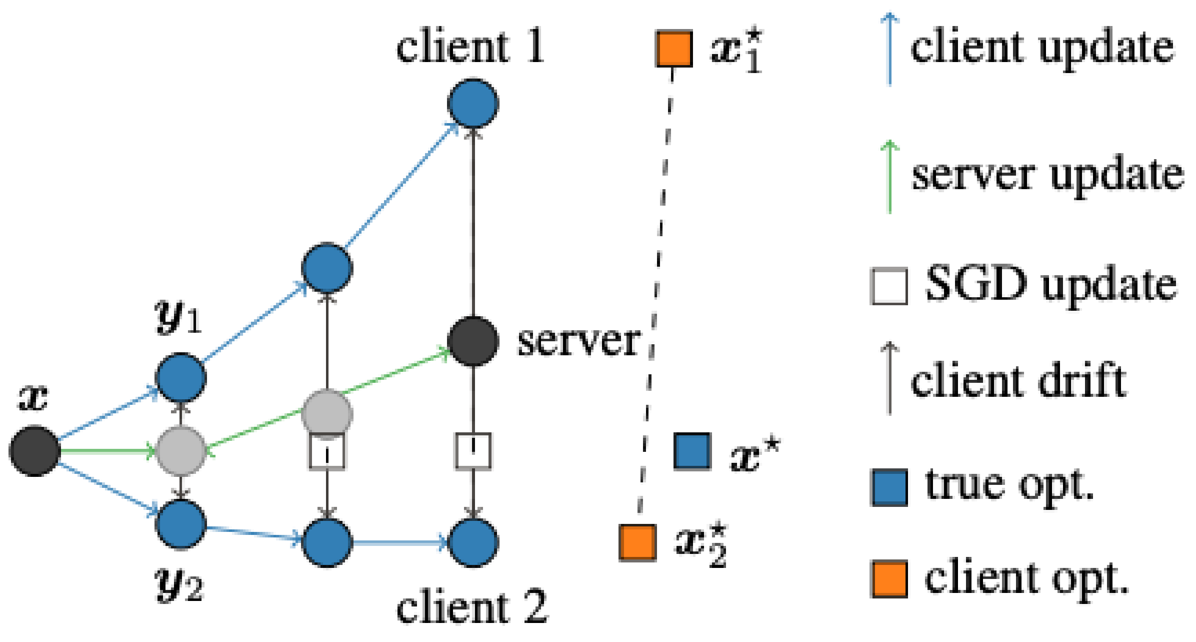


Figure 3: SCAFFOLD: An example Client-drift for 2 clients with 3 local steps. The server update moves away from the true opt.[9]

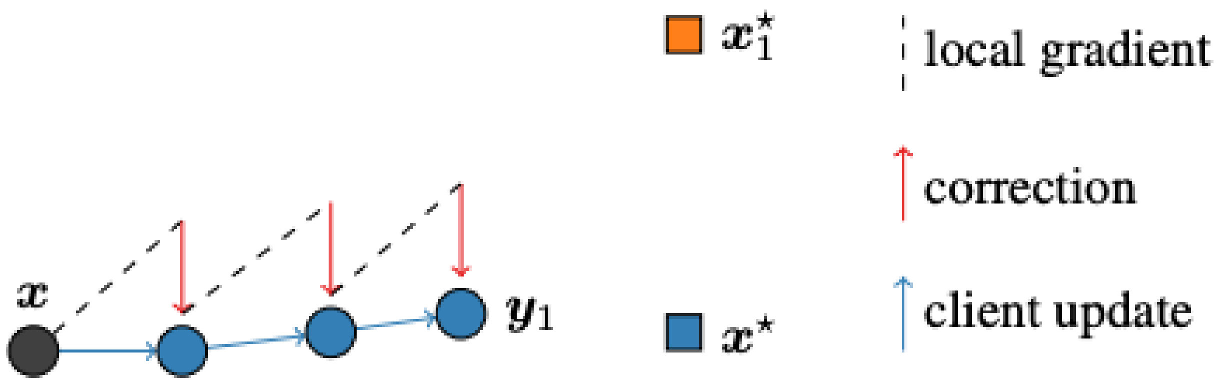
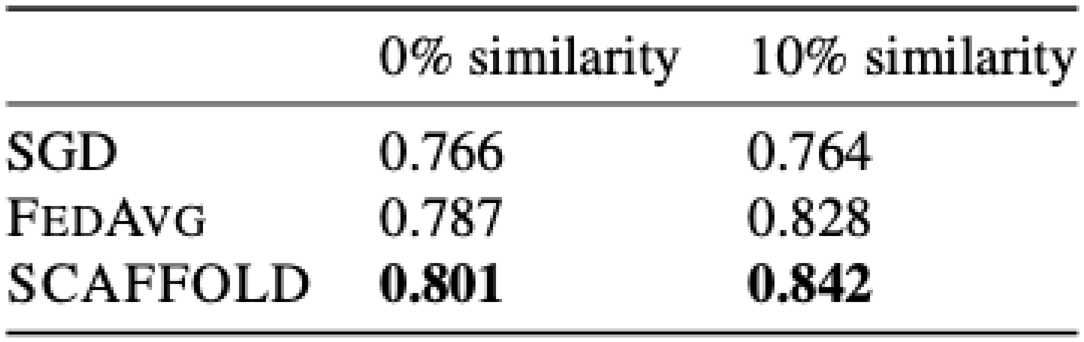


Figure 4: An illustration of how the correction causes the update to converge faster and tighter towards true optimum x*.[9]



	0% similarity	10% similarity
SGD	0.766	0.764
FEDAVG	0.787	0.828
SCAFFOLD	**0.801**	**0.842**

Figure 5: Test accuracy after 1000 rounds with 2 FC-NN on EMNIST dataset trained with 25 steps.[9]

FedAvg [12][11] and FedProx[10]. SCAFFOLD works extremely well and converges faster even if the similarity is around 0% and performs better than SGD proving that SCAFFOLD outperforms SGD in heterogeneous data. Surprisingly, SCAFFOLD gets faster with an increase in similarity whereas SGD is indifferent to any changes in similarity. Another edge over SGD is that with client sampling, there isn't a significant drop in convergence rate. Further research work can be done on real world visual data[5] for further experimentation. Even with 20% of client sampling per round, SCAFFOLD outperforms SGD in situations with and without any data similarity.

5 Evaluation

Method	FedAvg	FedProx	FedMA	FedAwS
Final accuracy	86.29	85.32	87.53	86.3

The above table is the measure of accuracy of the algorithms on CIFAR-10 dataset. From the table, we can clearly observe that FedMA seems to have the best accuracy. However, from the observation of SCAFFOLD results (as in Figure 5), it seems to outperform the other models, arguably behaving as the best approach to overcome the challenges of Federated Learning. [16] [7] [3]

6 Conclusion

In this summary, we have explored the concept of federated learning in detail. Federated learning is a statistical concept that trains data locally in edge devices and uses the result of training to improve the quality and accuracy of the results. We have seen the various challenges of federated learning like heterogeneity, communication overhead and privacy concerns. We also went through important approaches that are designed to overcome these challenges. Finally, we have compared the performances of these different algorithms from the available resources.

References

[1] Federated learning wikipedia page: https://en.wikipedia.org/wiki/federated_learning.

[2] Luca Corinzia and Joachim M Buhmann. Variational federated multi-task learning. *arXiv preprint arXiv:1906.06268*, 2019.

[3] X Yu Felix, Ankit Singh Rawat, Aditya Krishna Menon, and Sanjiv Kumar. Federated learning with only positive labels. *arXiv preprint arXiv:2004.10342*, 2020.

[4] Kevin Hsieh, Amar Phanishayee, Onur Mutlu, and Phillip Gibbons. The non-iid data quagmire of decentralized machine learning. In *International Conference on Machine Learning*, pages 4387–4398. PMLR, 2020.

[5] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Federated visual classification with real-world data distribution. *arXiv preprint arXiv:2003.08082*, 2020.

[6] Georgios A Kaissis, Marcus R Makowski, Daniel Rückert, and Rickmer F Braren. Secure, privacy-preserving and federated machine learning in medical imaging. *Nature Machine Intelligence*, pages 1–7, 2020.

[7] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020.

[8] Jeffrey Li, Mikhail Khodak, Sebastian Caldas, and Ameet Talwalkar. Differentially private meta-learning. *arXiv preprint arXiv:1909.05830*, 2019.

[9] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

[10] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 2018.

[11] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*, 2019.

[12] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data, 2017.

[13] Xingchao Peng, Zijun Huang, Yizhe Zhu, and Kate Saenko. Federated adversarial domain adaptation. *arXiv preprint arXiv:1911.02054*, 2019.

[14] Nicola Rieke, Jonny Hancox, Wenqi Li, Fausto Milletari, Holger Roth, Shadi Albarqouni, Spyridon Bakas, Mathieu N Galtier, Bennett Landman, Klaus Maier-Hein, et al. The future of digital health with federated learning. *arXiv preprint arXiv:2003.08119*, 2020.

[15] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. *Advances in neural information processing systems*, 30:4424–4434, 2017.

[16] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. *arXiv preprint arXiv:2002.06440*, 2020.

[17] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. Dba: Distributed backdoor attacks against federated learning. In *International Conference on Learning Representations*, 2019.