International Journal of Multiphysics
Volume 19, No. 1, 2025
ISSN: 1750-9548

Privacy-preserving based on federated learning with a case study on face recognition

Alireza Fathi[1]*

[1]Department of Computer Engineering, Faculty of Engineering, North Tehran Branch, Islamic Azad University, Tehran, Iran.
*Corresponding author

Abstract

Federated learning is a new machine learning technique that trains an algorithm on decentralized edge devices or servers containing local data without exchanging them. Federated learning provides a solution to enhance the security and privacy of users. This research aims to improve machine security and minimize the error rate. The security of face recognition and domain changing in federated learning are investigated and the existing challenges are addressed. Finally, two separate codes with and without TensorFlow were implemented. A special file was considered for global settings of parameters such as encryption status, timeouts, number of clients, client failures, simulated noise, etc. and the results were extracted. The TensorFlow library was modified for use in federated learning. Also, the number of users, unbalanced input data, data distribution in domain changing, low-speed communications in modeling, computational ability of edge devices or clients, model convergence time, the effect of encryption algorithms on the final results, the impact of adding private noise in the implemented algorithm, the effect of the epsilon parameter in the implemented algorithm were investigated. It was found that although the solution of the generative adversarial network (GAN) is good for solving the domain-changing problem, it does not meet the security requirements. Subsequently, adding differential privacy solved the domain-changing problem and security issues. In homomorphic encryption, the security of hashing codes and their impact was investigated. According to the results, although the encryption type flag can be changed, the state of private and public keys should be available to users. Finally, the serialization of modules was tested. Using cryptographic modules, differential privacy modules, GAN modules, multi-party computation (MPC) modules, and cumulative modules leads to the resolution of domain adaptation and change problems, prevention of repeated training, and solving the security problem. By applying the federated learning algorithm to face image data, the results were compared with the FedAvg and FedFace algorithms. The comparison result proved the greater flexibility of our algorithm than the existing algorithms.

Keywords: Face recognition, Federated learning, Security, Privacy.

Introduction

Over the past few years, face recognition has been widely studied by researchers in the computer vision and artificial intelligence domains. One of the critical issues in face recognition is the public concern about data privacy. Recently, the federated learning approach has been proposed by researchers to preserve privacy. This method can train a model collaboratively without sharing data between the parties. However, the full potential of federated learning in face recognition has not yet been fully realized.

Federated learning (1)(2) is an alternative to conventional machine learning that does not collect data. Parts of the algorithm that touch the data are transferred to the users' computers. Using their local data to improve the model, the users help train the model. Instead of sharing the data, the users only send these abstract improvements to the server. This approach is highly flexible and privacy-friendly. The improvement of mobile applications is particularly prominent in this regard.

Recent studies (3)(4) have shown promising results in using domain adaptation techniques to help analyze heterogeneous data, especially in the field of medical image analysis (5)(6). Federated learning, combined with domain adaptation methods in medical image recognition, extracts reliable and robust answers from imaging data (7).

The key objective of domain adaptation is to transfer the learned knowledge from one domain to the target domain. Subsequently, the trained model is refined on the data to adapt to different target domain data. Unsupervised domain adaptation methods have been widely studied in machine learning (8)(9)(10)(11)(12). However, these efforts cannot meet the requirements of federated configurations, especially when the data is stored locally and cannot be shared. The need to access both the source and target data hinders the application of adaptive approaches in federated learning domains. The domain adaptation approach in federated learning has been recently proposed (13)(14)(15)(16). Liu (17) introduced federated transfer learning to transfer the learned knowledge without leaking user privacy. Sharma et al. (18) enhanced federated transfer learning (17) by using an efficient MPC protocol called SPDZ to reduce the execution time and communication overhead. Peterson et al. (16) proposed private federated learning based on domain adaptation techniques. The proposed method first builds a general model based on differential privacy and then adapts the general model to each user's domain. Peng et al. (15) proposed adversarial federated adaptation techniques to address domain changing in a federated learning system. However, privacy issues were not addressed in the proposed method. In addition, it is not easy to extract target labels in the real-world model, and there are concurrent instances between the source and target domains. Song et al. (13) introduced an unsupervised domain adaptation method called PPUDA in the form of a distributed model that transfers learned knowledge without violating privacy. Bonawitz et al. (19) introduced a federated deep learning approach using a secure aggregation protocol based on SMC to protect individual model updates. Although the central server cannot explicitly access any local updates, it can observe the exact aggregation results in each epoch. The advantage of this approach is that it preserves the initial accuracy and guarantees privacy. However, the secure aggregation protocol imposes significant communication overhead. The key challenge of SMC-based FL methods is to improve computational efficiency. This is because significant computational resources are required to complete a training epoch in FL frameworks (20). Essentially, federated learning provides a framework for jointly training a global model using data stored in separate clients. However, according to recent studies, FL does not always guarantee sufficient privacy (21). This is because model parameters (weights or gradients) may leak sensitive information to malicious adversaries, leading to profound privacy violations (22).

Several privacy mechanisms can be configured and used. Differential privacy (23) is a mathematical privacy framework that incorporates the concept of risk when an individual's information is included in a dataset. The Laplacian mechanism uses random noise based on a symmetric Laplacian distribution. In addition to the private differential noise added by each client, the framework also supports the addition of distributed differential privacy noise. In distributed differential privacy, each client contributes a portion of the total differential privacy noise. The sum of all these individual contributions results in the differential privacy noise.

This research aims to study privacy-preserving face recognition based on federated learning. The main goal of this research is to simulate and provide a lightweight and efficient Python framework that allows clients to simulate secure federated learning while preserving privacy. Each client agent is able to interact with other client agents to generate shared keys before simulation.

Federated learning architecture

The main goal of this research is to simulate and provide a lightweight and efficient Python framework that allows clients to simulate secure federated learning while preserving privacy. In the simulation, each client corresponds to an instance of the client class. Each client agent is able to interact with other client agents to generate shared keys before simulation. Once a centralized system is configured, client agents communicate with an instance of the ServerAgent class. The server agent does not train its models. However, it executes the federated learning algorithm. The server agent has several responsibilities, such as requesting weights from clients, averaging the weights, and returning the federated weights to clients. Many applications only need one server agent. With some modifications, the framework can handle any number of server agents (Figure 1).

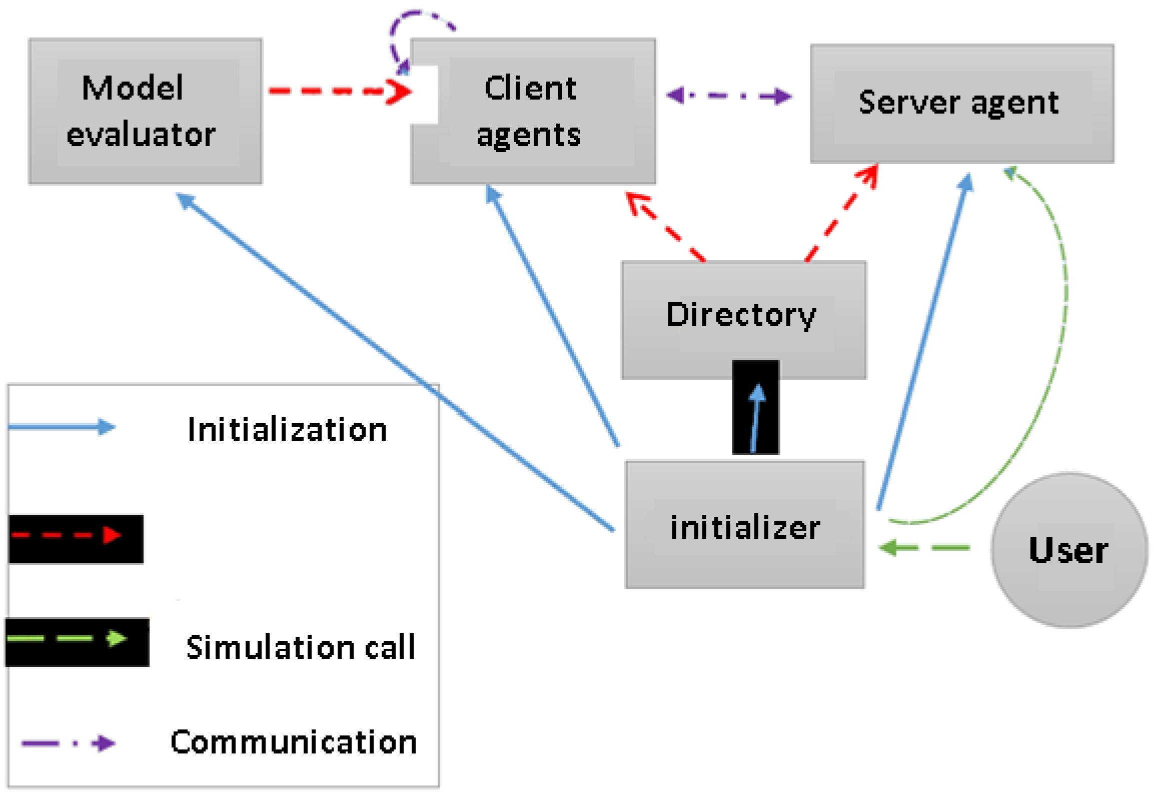

Figure 1: System graph containing relationships between important nodes in the simulator

Configuration and features

The simulator can be configured in various ways to simulate different scenarios with the configuration parameters available in py.config. The configuration parameters are described in Table (1).

Table 1: Configuration parameters

USE_SECURITY	If this parameter is true, clients perform a Diffie Hellman key exchange in the offline part of the simulation to generate encryption keys. This parameter has no effect on the accuracy of the participation.
USE_DP_PRIVACY	If this parameter is true, clients will add noise with the specified parameters to the configuration file.
SUBTRACT_DP_NOISE	If this parameter is true, clients will reduce the noise added to the model. If this parameter is false, clients will use the federated model computed by the server.
CLIENT_DROPOUT	If this parameter is true, the client is excluded from the simulation process. The simulation process continues without the client.
SIMULATE_LATENCIES	If this parameter is true, the system can simulate the completion time of each step of the protocol with the communication delay defined by the user in the configuration. If this parameter is false, such information will not be displayed.
USING_CUMULATIVE	If our data partitioning module is used, this flag is useful for testing data options. If this parameter is false, the dataset at each iteration contains only new data. This makes perfect sense for Algorithm 1. If this parameter is true, the dataset size at each iteration increases by the amount of new data.

	This configuration makes perfect sense for Algorithm 2, where the weights of the ith iteration are not used in generating the weights of the i+1th iteration.

In addition, py.config allows system designers to configure several other parameters such as the number of clients, custom ϵ size, and data sizes for each client, and secure random exploration for repeatability. For face recognition, optimization, and federated learning security, this research uses various datasets including the FER2013 dataset in Challenges in Representation Learning: Facial Expression Recognition Challenge in Kaggle, MNIST, and KDDCUP19.

First, we investigated the security of face and image recognition in the form of federated learning and domain changing and extracted the results. Then, we addressed the existing challenges and implemented two codes with/without TensorFlow. The number of users, unbalanced input data, data distribution in domain changing, low-speed communication in modeling, computational ability of edge or client devices, model convergence time, the effect of encryption algorithms on the final results, the effect of adding private noise in the implemented algorithm, the effect of the epsilon parameter in the implemented algorithm were investigated. The significant results obtained are presented in the next section. In general, face recognition algorithms in the form of federation learning suffer from several challenges that can be partially resolved in customized code.

The flowchart of the proposed method is represented in Figure (2). The first code includes the implementation of federated learning in face recognition. This code is based on the TensorFlow library and 1 client. The Tensorflow library is installed as follows.

# Tensorflow federated package !pip install –

upgrade tensorflow_federated > /dev/null 2>&1

The results of the execution for each client are represented in Figure (2).

Due to the need for FL customization, the second code addresses the specific cases and some disadvantages of Tensorflow (6). Privacy concerns remain prominent. This is because information about the training dataset is leaked from models trained with specific weights or parameters. Therefore, developing centralized learning algorithms to train highly accurate models is crucial for privacy preservation. Setting up a shared training platform that guarantees security and privacy is a time-consuming process. Also, numerous configurations and parameters need to be manipulated. Several metrics such as accuracy, time, and privacy for clients are considered.

Figure 2: Flowchart of the proposed method

To provide the necessary security guarantees, a combination of algorithms (Diffie-Hellman key exchange and pseudo-random generators, differential algorithms, and delay management) is used, which gives us favorable results regarding accuracy, time, and security. According to this method, clients only need to communicate with each other once at the start of the simulation. According to this method, clients only need to communicate with each other once at the start of the simulation. This process ensures the security of all iterations and consequently reduces the amount and time of client-client communications.

Results

In the evaluation phase, the clients were configured and trained on the dataset (24) with eight iterations. In this study, four experiments (including accuracy and quality comparison and client number trade-off, privacy constraints, decentralized federated training, and real-world delay simulation) were conducted as follows.

Experiment 1: Accuracy and quality comparison and client number trade-off

The client simulator enables the evaluation of the decrease/increase in model accuracy after participating in federated learning. For this reason, the average accuracy of the client model is compared with the global model accuracy at each iteration. Algorithm (2) is used for this purpose. To calculate the client accuracy at the ith iteration, its weight is considered without differential privacy. This is because the scenario concerns clients not participating in federated learning. Therefore, there is no need to add differential privacy noise.

Figure (3) compares the average accuracy of the three clients with the federated accuracy for different values of ϵ. A smaller value of ϵ corresponds to less noise. As expected, the accuracy of the federated model improves with increasing ϵ. However, there is no significant difference between ϵ = 1 and ϵ = 8. Also, the federated accuracy for the line ϵ = 1.0 increases faster than the average client accuracy. The amount of noise added per client is smaller than the amount of data per client (thirty samples increase per iteration). In the next example, we simulate a larger number of clients. In this example, we use the KDDCup99 [331] dataset and Algorithm (3).

Figure (4) represents the result of varying the number of clients and ϵ. As expected, the accuracy of the federated model improves with increasing ϵ due to the addition of less noise. Also, the accuracy increases with increasing the number of clients. This behavior was predictable. The reason is that the accuracy of the federated model increases as more clients participate in the model. Also, the convergence speed increases as the number of parties increases.

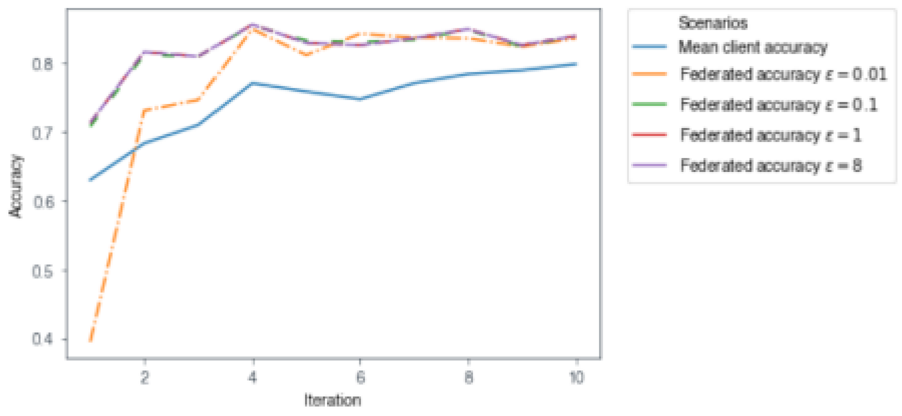

Figure 3: Average client accuracy on test data versus federated model accuracy for different values of ϵ. Clients follow Algorithm 2.

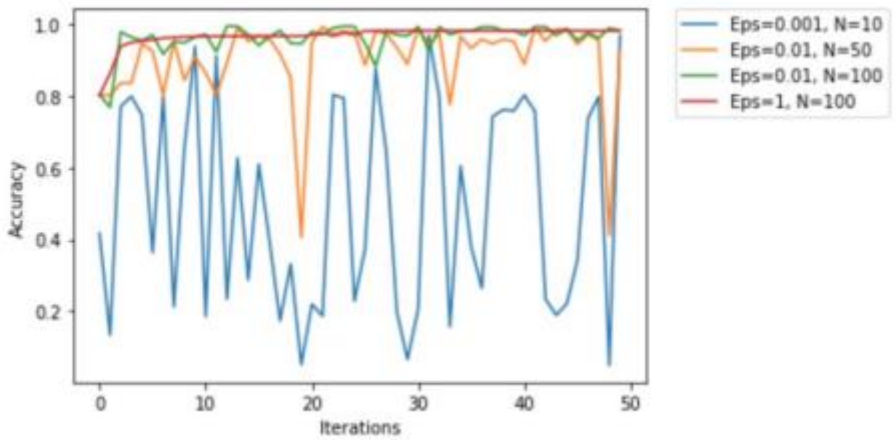

Figure 4: Accuracy versus iteration for different values of ϵ and number of parties

Experiment 2: Privacy constraints

In this experiment, each client can have different amounts of data and privacy requirements. Suppose Clients 1 and 2 have 150 data points and ϵ=1. However, Client 3 has 250 data points and has a stricter privacy requirement (ϵ=0.1). Clients 1 and 2 decide whether to cooperate or not with each other, whether to include or not include Client 3, and whether to participate in federated learning.

In Scenario 1, no clients participate in federated learning. Therefore, their accuracy is the accuracy of their own model. In Scenario 2, Clients 1 and 2 participate in learning, unlike Client 3. This is because Client 1 has $\epsilon = 1/3$ while Clients 3 and 2 have $\epsilon = 1$.

In Scenario 3, all three clients participate in federated learning and use the privacy requirements of Client 3, which is $\epsilon = 0.1$.

In Scenario 4, all three clients participate in federated learning. However, Clients 1 and 2 use their own privacy requirements ($\epsilon = 1$) and Client 3 uses its own privacy requirements (ϵ=1.1).

Table 2: Accuracy of each client using Algorithm 2 in four different scenarios

Client 3	Client 2	Client 1	Clients / Scenarios
0.823	0.793	0.750	Scenario 1
-	0.810	0.806	Scenario 2
0.813	0.800	0.767	Scenario 3
0.850	0.830	0.827	Scenario 4

Table (2) shows the accuracy of the clients in each scenario. These values were obtained using the configuration options of Algorithm 2. According to the table, Clients 1 and 2 do not benefit from including Client 3 in the simulation except under the condition that each client uses its own value of ϵ. In Scenario 4, each client benefits from participating in federated learning.

Experiment 3: Decentralized federated learning (without a server)

The simulator can be modified by changing the configuration parameters (config file). In the following example, a new type of agent is created capable of federated machine learning without a server. In this case, clients send their weights directly to other clients. Once the weights are received, the client can calculate the average of them to create a federated model. Figure (5) shows the results of training three clients with Algorithm 2 on thirty data points for seven iterations. Although the clients use a security protocol, they do not consider different privacy. This behavior is achieved by setting PRIVACY_DP_USE and SECURITY_USE to False and True, respectively. As shown in Figure (5), all clients clearly benefit from participating in federated learning.

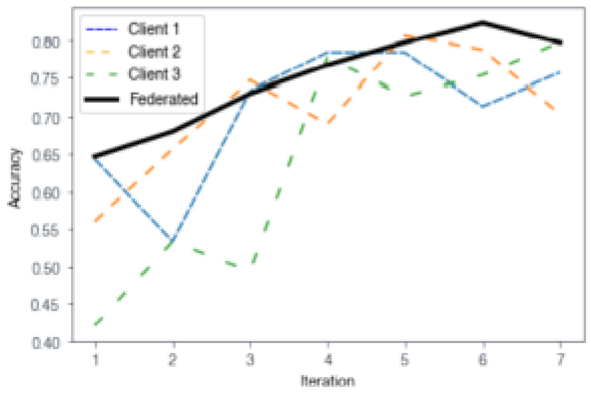

Figure 5: Simulation with modified Algorithm 2 without the need for a server

Experiment 4: Real-world delay simulation

In this experiment, the presence of clients in different geographical locations was simulated. Three clients were configured, located in Boston, Singapore, and New York. The client is closer to the client in New York. Thus, the following communication delays were considered: (1) Boston server: 1.1 seconds (2) Singapore server: 2 seconds, and (3) New York server: 0.1 seconds. Client delays can also be adjusted. Due to the lack of client-client communication during the online part of the simulation, these only apply to the offline part of the simulation.

Table 3: Simulated time to obtain federated weights in the outbound Singapore sample (farthest client) after the second iteration

Simulation time to receive federated weights (seconds)			
New York	Singapore	Boston	Location / Iteration
4.110	6.010	4.310	Iteration1
4.106	6.006	4.306	Iteration2
0.709	-	0.909	Iteration3

Table (3) represents the simulated time. This time is measured from the start of each iteration until the clients receive the federated weights from the server. The simulation time is included in each message between the clients and is available for each step of the protocol. Upon receiving the message, the time agent performs its logical operation. The agent then adds the logical operation execution time to the simulated communication time for each client. This creates a new simulation time for the receiving agent. According to Table 1, the simulation times for Boston and New York are significantly lower in the third iteration. As soon as DROPOUT_CLIENT is set to True, clients can drop out. The SingaporeServer latency is the highest, meaning that once Singapore drops out at the end of the second simulation, other clients receive the federated weights faster. This is because the server does not need the weights of the Singapore client. Similarly, the time required to calculate the weights in each iteration is displayed by the simulator. In this study, a simulator for privacy-preserving and secure federated learning is presented. The primary features of the system include delay simulation, robustness against client exit/failure, support for server-based and serverless federated learning, and configurable privacy parameters.

Conclusion

In the first experiment, the positive effect of the accuracy/quality and trade-off of the number of clients in the proposed customized module was demonstrated. The accuracy of the federated model is enhanced by increasing ϵ due to the addition of less noise. Also, increasing the number of clients leads to increased accuracy. This is because increasing the number of clients participating in the federated model increases the accuracy of the federated model. Also, the convergence speed increases with increasing the number of parties. Each client benefits from participating in the customized module and experiences higher accuracy than traditional methods. Privacy constraints are addressed in the second experiment. Each client can have different amounts of data and privacy requirements. In the third experiment, each client in the proposed customized module can be a server for other clients. In this situation, clients send their weights directly to other clients. According to the results, all clients benefit from participating in federated learning. The effect of delays was addressed in the fourth experiment. This time is measured from the start of each iteration until the clients receive the federated weights from the server. The simulation time is included in each message between clients and is available for each step of the protocol. Upon receiving the message, the time agent performs its logical operation. The agent then adds the logical operation execution time to the simulated communication time for each client. This creates a new simulation time for the receiving agent. Clients are also allowed to exit. When a client exits at the end of the second simulation, other clients receive the federated weights faster. This is because the server does not need to wait to receive the weight of the exited client. Similarly, the time required to calculate the weight in each iteration is represented by the modulus.

According to the results, evaluating solutions for modeling malicious threats where clients do not require protocols and direct adversarial attacks, as well as extending the federated simulator to real-time cluster environments and testing on big data, are recommended for future work.

References

1. J.Konečný, H.B.McMahan, F.X.Yu, P.Richtárik, A.T.Suresh, and D.Bacon, “Federated Learning: Strategies for Improving Communication Efficiency,” *Iclr*.2016, [Online].Available: http://arxiv.org/abs/1610.05492.

2. H.Brendan McMahan, E.Moore, D.Ramage, S.Hampson, and B.Agüera y Arcas, "Communication-efficient learning of deep networks from decentralized data," 2017.
3. Y.Yao, Y.Zhang, X.Li, and Y.Ye, "Heterogeneous domain adaptation via soft transfer network," 2019, doi: 10.1145/3343031.3350955.
4. X.Wang *et al.*, "Deep mixture of experts via shallow embedding," 2019.
5. C.Chen, Q.Dou, H.Chen, J.Qin, and P.A.Heng, "Synergistic image and feature adaptation: Towards cross-modality domain adaptation for medical image segmentation," 2019, doi: 10.1609/aaai.v33i01.3301865.
6. J.Yang, N.C.Dvornek, F.Zhang, J.Chapiro, M.De Lin, and J.S.Duncan, "Unsupervised Domain Adaptation via Disentangled Representations: Application to Cross-Modality Liver Segmentation," 2019, doi: 10.1007/978-3-030-32245-8_29.
7. X.Li, Y.Gu, N.Dvornek, L.H.Staib, P.Ventola, and J.S.Duncan, "Multi-site fMRI analysis using privacy-preserving federated learning and domain adaptation: ABIDE results," *Med.Image Anal.*, 2020, doi: 10.1016/j.media.2020.101765.
8. M.Long, H.Zhu, J.Wang, and M.I.Jordan, "Deep transfer learning with joint adaptation networks," 2017.
9. J.Hoffman, M.Mohri, and N.Zhang, "Algorithms and theory for multiple-source adaptation," 2018.
10. M.Long, Y.Cao, Z.Cao, J.Wang, and M.I.Jordan, "Transferable Representation Learning with Deep Adaptation Networks," *IEEE Trans.Pattern Anal.Mach.Intell.*, 2019, doi: 10.1109/TPAMI.2018.2868685.
11. M.Long, Z.Cao, J.Wang, and M.I.Jordan, "Conditional adversarial domain adaptation," 2018.
12. E.Tzeng, J.Hoffman, K.Saenko, and T.Darrell, "Adversarial discriminative domain adaptation," 2017, doi: 10.1109/CVPR.2017.316.
13. L.Song, C.Ma, G.Zhang, and Y.Zhang, "Privacy-Preserving Unsupervised Domain Adaptation in Federated Setting," *IEEE Access*, 2020, doi: 10.1109/ACCESS.2020.3014264.
14. C.Ju, D.Gao, R.Mane, B.Tan, Y.Liu, and C.Guan, "Federated Transfer Learning for EEG Signal Classification," 2020, doi: 10.1109/EMBC44109.2020.9175344.
15. X.Peng, Z.Huang, Y.Zhu, and K.Saenko, "Federated Adversarial Domain Adaptation," Nov.2019, Accessed: Sep.18, 2021.[Online].Available: http://arxiv.org/abs/1911.02054.
16. D.Peterson, P.Kanani, and V.J.Marathe, "Private Federated Learning with Domain Adaptation," Dec.2019, Accessed: Sep.18, 2021.[Online].Available: http://arxiv.org/abs/1912.06733.
17. Y.Liu, Y.Kang, C.Xing, T.Chen, and Q.Yang, "A Secure Federated Transfer Learning Framework," *IEEE Intell.Syst.*, 2020, doi: 10.1109/MIS.2020.2988525.
18. S.Sharma, C.Xing, Y.Liu, and Y.Kang, "Secure and Efficient Federated Transfer Learning," *Proc.- 2019 IEEE Int.Conf.Big Data, Big Data 2019*, pp.2569–2576, 2019, doi: 10.1109/BigData47090.2019.9006280.
19. K.Bonawitz *et al.*, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the ACM Conference on Computer and Communications Security*, 2017, pp.1175–1191, doi: 10.1145/3133956.3133982.
20. M.Sadegh Riazi, K.Laine, B.Pelton, and W.Dai, "HEAX: An architecture for computing on encrypted data," in *International Conference on Architectural Support for Programming Languages and Operating Systems - ASPLOS*, 2020, pp.1295–1309, doi: 10.1145/3373376.3378523.
21. J.Geiping, H.Bauermeister, H.Dröge, and M.Moeller, "Inverting gradients - How easy is it to break privacy in federated learning?," in *Advances in Neural Information Processing Systems*, 2020, vol.2020-Decem.
22. L.Zhu, Z.Liu, and S.Han, "Deep leakage from gradients," in *Advances in Neural Information Processing Systems*, 2019, vol.32.
23. C.Dwork, F.McSherry, K.Nissim, and A.Smith, "Calibrating noise to sensitivity in private data analysis," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2006, vol.3876 LNCS, pp.265–284, doi: 10.1007/11681878_14.
24. Lecun Yann, Cortes Corinna, and Burges Christopher, "THE MNIST DATABASE of Handwritten Digits," *Courant Inst.Math.Sci.*, 1998.