A study on Privacy-Preserving Federated Learning and enhancement through Transfer Learning

Robert Minea[1] **Kaitai Liang**[2] **Rui Wang**[3]
TU Delft
r.minea@student.tudelft.nl, kaitai.liang@tudelft.nl, r.wang-8@tudelft.nl

27th of June 2021

Abstract

Privacy in today's world is a very important topic and all the more important when sizeable amounts of data are needed in Neural Network processing models. Federated Learning is a technique which aims to decentralize the training process in order to allow the clients to maintain their privacy, while also contributing to a broader learning process. In order to allow parties that undertake similar tasks to share data between them, even if they don't follow the same feature representation or domain distribution, Transfer Learning is also used in order to augment the learning by sharing knowledge with the contributing parties. The name of this combination of techniques is Federated Transfer Learning. This paper aims to showcase the strengths and weaknesses of Federated Learning through a simple implementation while comparing different Federated Transfer Learning frameworks that can be used in order to enhance the capabilities of a simple federation of clients that are contributing towards the learning of a similar task.

Keywords— Federated Learning, Privacy-Preserving, Neural Networks, Transfer Learning, Knowledge Distillation, Decentralized Learning, Differential Privacy, Homomorphic Encryption, Secure Multi Party Computing, Artificial Intelligence

1 Introduction

Problems in the contemporary setting are becoming increasingly complex, and in order to facilitate answers to such a diverse palette of problems and questions, more intricate Artificial Intelligence algorithms had to be developed. This is how strategies such as Neural Networks [1] appeared, in which the algorithm learns how to solve flexible problems by itself. In order to do so, plenty of training data must be used in the learning process, and sometimes problems such as government regulations and policies might stand in the way of attaining a diverse enough training dataset. [2]

Federated Learning provides a feasible solution by decentralizing the sample data and involves many parties in the learning process, each contributing in this development, while keeping the data private.[3] One of the techniques that is used in order to mould the model and datasets such that knowledge can be reused by other parties involved in this process is Transfer Learning, fusing into a powerful combined strategy called Federated Transfer Learning. [4]

Even though Federated Learning provides a way of preventing communication overhead by not moving large datasets between the clients through decentralized learning, it also presents itself with many facets that can be exploited by adversaries whose goals are malicious with respect to the performance and security of the final trained model.

In order to properly emphasize the facets of Federated Transfer Learning, its advantages, possible weaknesses and implementation strategies, this work will be organized as follows. Section 2 will present the main research methods that were used in order to achieve a proper understanding of the multi-sided Federated Transfer Learning environment. Section 3 will go through the simulation of an algorithm implementing a basic version of a federation. In close link to the simulation, the results will follow in Section 4 together with an analysis of the frameworks that enhance Federated Learning through Transfer Learning techniques. Since careful consideration should be given to the integrity of the research process, Section 5 contains important mentions about Responsible Research and responsible techniques used in general. Section 6 will focus on discussions on various topics regarding Federated Transfer Learning. The final chapter is the conclusion in which the final statements following the research project will be made, together with possible future work or ideas that can be applied.

2 Methodology and Background Knowledge

Since following the horizontal iterative process of slowly discovering and understanding more about a subject is a recommended way of approaching a novel subject, this section will follow the progress that was undertaken in order to understand the topic of Federated Transfer Learning.

2.1 Federated Learning

The first step of understanding the main concepts of Federated Learning was browsing through scientific and internet articles in order to grasp the focus of this subject. One of the most widely known sources in the field was published on the Google AI blog [3]. In this article which treats the Federated Learning concept [5], the Google Keyboard is presented, and the mechanism behind it is explained. As mentioned in the introduction on the topic, the training of the keyboard is decentralized, relying on the data stored on each of the devices, without sharing it. The model training is done on each device and after every round of training the central server garners all the trained models and aggregates the result into a new global model. This model will be sent back to the clients for the cycle to repeat, as showed in Figures 1 and 2.

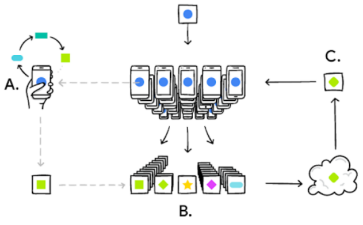

Figure 1: The continuous learning process using the data on the user devices. [3]

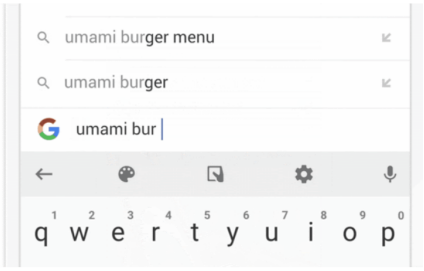

Figure 2: Image of a project implementing the federated mechanism, the Google Keyboard (or Gboard). [3]

2.2 Data-Dependent Types of Learning

As expected, the datasets which are used in the learning process can be very different, either in the feature or sample space. Taking in consideration these differences the following divisions of learning can be made:

- **Horizontal Federated Learning**: This learning technique requires the users of the network to share the same feature space. The learning model is sent by all users to a central server which aggregates the result and returns the new aggregated model to all the users in the network for the next round of learning.
- **Vertical Federated Learning**: This kind of learning is used when the parties contain samples about the same subject but with different features. Only the common samples from all the parties are going to be treated in the learning process of the remote clients.
- **Federated Transfer Learning**: Another kind of Federated Learning which involves "Transfer Learning" techniques is based on training the model on some samples, "transferring" the knowledge extracted from these samples and applying it on other samples from parties which are considering a different feature space and data distribution in their dataset. [4] Formally, transfer learning aims to improve the efficiency of executing the target task $\mathcal{T}_T$, which contains an already existing training dataset $\mathcal{D}_T$ through usage of the knowledge learned in the source task $\mathcal{T}_S$ by the use of the dataset $\mathcal{D}_S$ (where $\mathcal{D}_S \neq \mathcal{D}_T$ and $\mathcal{T}_S \neq \mathcal{T}_T$). [6]

2.3 Security and Privacy

As the federated context introduces a brand new method of training Neural Networks, satisfying the privacy issues that come with current regulations about user privacy is of great importance. Federated Learning is a technique which can be attacked from different surfaces than a centralized Neural Network. For a regular Neural Network, the focus of the attacks is to find the limits of the trained model in the inference phase, while in the federated context, a malicious user seeks to corrupt the network mainly during the training phase. [7] Two types of adversaries can be discerned in the federated setting:

- **Semi-Honest Adversaries**: This kind of users do not stray from the protocol given by the system that they are part of, but they are curious, and by wanting to know more they try to reach information which is not always made transparent by the protocol.
- **Malicious users**: The malicious users might not even have any benefit from approaching the protocol in a malevolent way, this is why they are very dangerous since they might desire only the promotion of chaos, if not backed by ill-natured motives. [8]

Some of the following attacks can be made on the network [9] [10]:

- **Model Corruption**: As mentioned above, when the context of learning is a federation of clients connected to a central server, we can encounter this type of attack in which an ill-intentioned user can change learning parameters that will affect the performance of the final data model.
- **Data Poisoning**: This kind of attacks have the training dataset as target. A malicious user can either change the labels of some of the training samples, or it can insert watermarks into multiple samples in order to generate triggers for specific behaviour, which only the attacker is aware of.
- **Data Privacy Attacks**: The advantage of having a federation of learning clients is that the dataset is not passed, thus many privacy regulations are being respected, but recent research [11] shows that certain details about the training dataset can be extracted from a trained model that memorizes some key features from the training dataset, compromising the data privacy in cases where it should actually not be disclosed. Data Privacy attacks are further split into attacks such as, Model Inversion and Reconstruction attacks which aim to discover

the training set through access to the model and Membership-inference attacks which can also pose a problem to data privacy, they aim to discover which samples are part of a dataset through inference using a black-box model.

In order to protect the federation against possible breaches in security and privacy, various techniques have been devised in order to counteract or alleviate the possible attacks, as Federated Learning allows for a wider range of breaches since communication must be present between the parties, thus giving possibly malicious third-parties more surfaces from which an attack can be executed.

- **Secure Multi-Party Computing**: As the title of the security technique already infers, multi-party computing is a technique which involves the computation in parts of a final function f, in which each participant holds a secret part of the input. The result is of form $f(x_1, x_2..x_n) = y_1, y_2..y_n$, in which each client holds a secret share of the final result. Multiple implementations of multi-party computation have been proposed, among those are Oblivious Transfer [12], which uses encryption and seeks to allow an exchange of parts of information from one party, while keeping the sending party and receiving party unaware of the parts of the message that they are processing. The usage of Beaver triples [13] also allows the execution of secret sharing, an offline generator must be created to generate random tuples that are used in the secret sharing process by the parties.
- **Homomorphic Encryption**: The main principle of homomorphic encryption is to allow computation on encrypted data such that the final decryption returns the same result as if the operation has been done on the plaintext, not on the cyphertext. Multiple techniques have been implemented along the years, like the one proposed by G. Craig [14], which allows unlimited additions and multiplications on the cyphertext. The flexibility of the encryption technique usually comes with an unwanted additional computation overhead which compromises the performance of the algorithms. Three groups of homomorphic encryption can be distinguished based on their capabilities. Partial homomophic encryption allows either addition or multiplication for an unlimited number of times. Somewhat homomorphic encryption allows for both addition and multiplication for a limited number of times. Due to noise being added in order to ensure security at every iteration, an upper sensitivity level is reached in which the cyphertext can not be decrypted correctly anymore. The final type is fully homomorphic encryption which allows unlimited additions and multiplications at the cost of expensive computation. According to the formal definition of Homomorphic Encryption, any type of encryption whose property allows $\text{Enc}(m_1 *_M m_2) \leftarrow Enc(m_1) *_C Enc(m_2)$ can be called homomorphic. In this definition, $\leftarrow$ shows that the left part can be computed from the right part without any necessary intermediate decryption.
- **Differential Privacy**: Differential Privacy relies on a simple principle, not allowing an outside party to find out specific information about database samples by introducing a small random factor in either the dataset samples, the model weights or during the training process. Differential privacy is a technique that ensures protection against membership-inference attacks and backdoor model poisoning attacks, since the final result of the calculations does not reveal sensitive information about one of the samples used in the calculation [15]. Unfortunately this technique displays quite a sizeable issue, as increasing the noise added to the data will ensure better privacy at the cost of final calculation accuracy. Thus a model in which differential noise is added will take longer to train, and the final result will probably not reach the same level of accuracy as normal training.

In order to check a new framework, the information from this section was used in order to understand the techniques used by the network, and determine the weaknesses and strengths of each Federated Transfer Learning Framework.

3 Implementation

In order to experience the benefits of decentralized learning, an implementation of a federation has been made according to the standard presented in [5]. The implementation was used to simulate how a federation works on a very small scale, and has been written according to a course on Federated Learning [16]. Creating the implementation of a Neural Network from the ground up is a time-consuming task, so in order to streamline the development process, the PyTorch [17] library for Machine Learning has been used, together with the PySyft library from OpenMined [18], that facilitates the application of Federated Learning among the Neural Networks present on each machine.

Moreover, another implementation has been created in order to demonstrate the accuracy and performance sacrifices that must be made when implementing Differential Privacy. The Opacus library [19] has been used in order to add random noise during the backpropagation process of a Convolutional Neural Network trained on the entire MNIST dataset.

3.1 Architecture

Virtual machines have been created in the Google Cloud environment for both the central server and the clients participating in the learning federation. Each machine runs an instance of a Jupyter Notebook that uses the Duet library of the PySyft service in order to create connections between the notebooks. The Convolutional Neural Network distributed on each of the client machines is built according to the model used in the PyTorch MNIST example, and includes two bidimensional convolutional layers, two fully connected layers, rectified linear units as activation function and two dropout layers used in regularization. A step function is used to reduce the learning rate by by 5% every epoch, starting from η=1.

4 Results and Framework Comparisons

In order to check the efficiency boost of Federated Learning with multiple clients against a simple centralized method, experiments have been made using the Architecture specified in Section 3.1 with training sets containing 1280 randomly chosen samples on each round of learning from the MNIST dataset on each client device.

	1	2	3	4	5	6	7	8	9	10
Fed	9.85	85.86	92.29	94.35	94.89	95.73	96.26	96.61	96.77	97.06
Net	80.14	93.11	94.47	95.08	95.58	96.44	96.26	94.88	95.97	97.01

Table 1: Table containing the evolution of test set accuracy, for a Federation of 5 clients (Fed) compared to a single Neural Network (Net).

The graphic shows the evolution of the test accuracy (executed on 10000 MNIST test samples) for the execution of a single client against a federation of clients. As expected, in the beginning, the federated algorithm only reaches a small test accuracy due to the averaging of 5 weak models, but the accuracy boost quickly rises, as each of the clients contributes better models, discovering exclusive specific features from their individual datasets, which are then aggregated into a single model. It can be said that a federation can reach a better accuracy than a simple Neural Network, if it is allowed to run for a sufficient number of rounds, as seen in the plot from Figure 4, the accuracy is steadily improving, while the performance of the singular model is wavering.

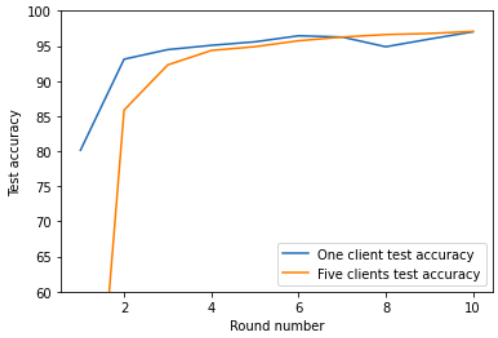

Figure 4: Graphical evolution of the test accuracy for the models corresponding to Table 1

Another experiment has been made on a Convolutional Neural Network created to recognize the digits of the MNIST dataset to check the impact of Differential Privacy (pictured in Figure 5). The advantage of using Differential Privacy lies in better protection against Membership-Inference, Reconstruction attacks and Backdoor attacks, but it comes at the price of accuracy and performance. While the regular model reaches an accuracy of 99.2% on the 10000 test samples of the MNIST dataset, the implementation using Differential Privacy has a very rought accuracy at the beginning and comes to its roof performance at 91.51% test accuracy. The deviation from the Gaussian distribution from which noise is added can be changed in order to generate less noise, but will come at the cost of lower privacy.

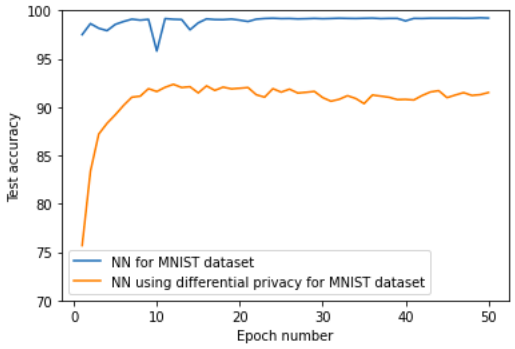

Figure 5: Graphical evolution of the test accuracy for a Neural Network with and without Differential Privacy noise added.

4.1 Transfer enhancement using the studied framework techniques

The implementation follows a case of learning in which all the samples' feature vectors are of the same length and are representations of the pixel values of the images from the MNIST dataset. This is an ideal case, but in a real world scenario, in which different companies from the same field are part of a learning federation, the domain distribution, feature representation or client models might be very different, requiring the use of Transfer Learning. Thus different papers present various frameworks for Federated Transfer Learning in order to solve the above mentioned issues. Firstly we will talk about the time-complexity of Neural Network training, followed by a brief introduction of the first paper proposing the Federated Learning framework and a sequence of papers that

introduce various techniques to improve the federation through transfer mechanisms. To finalize, a general comparison between the frameworks is presented.

Time-complexity calculation: After looking into the showcased papers' algorithms, time complexities for the algorithms have also been calculated, not taking in consideration the transmission times as it is heavily dependant on the network architecture. In order to facilitate shorter complexity formulas, we are going to shorten the training and inference complexities by saying that they are dependent on the local model, thus being $\mathcal{O}$(train) and $\mathcal{O}$(infer). The true complexity for the mentioned short versions of the inference and training are dependent on the number of layers and their number of nodes. Since a forward pass by one layer can be implemented through matrix multiplication, we have the following complexity calculation:

$$F_j = W_{ji} \cdot I_i \tag{1}$$

$$A_j = f(F_j) \tag{2}$$

Since W_{ji} is the matrix representing i weights and j neurons, and I_i is a feature vector of one sample, we can say that the calculation to the final activation result A_j is of complexity $\mathcal{O}(i \cdot j)$, if n training samples are involved in this forward feed, the final result would be $\mathcal{O}(n \cdot i \cdot j)$. Thus a final complexity for training in a k-layered neural network with n training samples, in e epochs:

$$\mathcal{O}(train) = \mathcal{O}(e \cdot n \cdot (l_1 \cdot l_2 + l_2 \cdot l_3 + ... + l_{k-1} \cdot l_k)) \tag{3}$$

4.1.1 Communication-Efficient Learning of Deep Networks from Decentralized Data [5]

Being the vanilla version of Federated Learning, this paper created by Google researchers is one of the first implementations of a data federation, it presents the main principle of Federated Learning, together with central server aggregation done using model averaging. It is a framework that can be applied to a Horizontal Federated Learning setting, but it is a proper starting framework in the process of developing federated algorithms. The goal of a federation is to minimize the general loss function, knowing that F_k is the loss function of client k, K is the total number of clients and n_k is the number of local dataset samples:

$$\min_w f(w) \qquad where \qquad f(w) = \Sigma_{k=1}^{K} \frac{n_k}{n} \cdot F_k(w) \tag{4}$$

$$F_k(w) = \frac{1}{n_k} \Sigma_{i=1}^{n_k} f_i(w) \tag{5}$$

The algorithm, which can be found in Appendix A, is round based, all the clients are training their local model for a given number of epochs E in parallel. Following the training process, the models will be passed to the central server in order to create an aggregated model through averaging the model parameters. The newly created global model will be sent back to the clients for a new round of learning.

Server time complexity:

$$\mathcal{O}(K \cdot M) \tag{6}$$

Client time complexity:

$$\mathcal{O}(train) \tag{7}$$

Message size complexity:

$$\mathcal{O}(M) \tag{8}$$

Symbols: K - number of clients (can be ignored if parallel); M - local model parameters and settings.

Security and Privacy
Since it follows the main principle of federated learning, this framework keeps the dataset of each client private, only the local client models being passed. Although the datasets are being kept private, information about them can still be found through model inversion, and many kind of attacks such as data poisoning, model poisoning or membership-inference can still be applied to this framework, as it does not offer any way to secure the data against a malicious client or a malicious server.

4.1.2 Selective Federated Transfer Learning using Representation Similarity [20]

A very straightforward and bold approach is proposed in the paper presenting this first Federated Transfer Learning Technique, which uses model similarity as a metric of choice for selecting which model to use as source during the training process. The algorithm is based on model similarity, which is calculated according to a CKA (central kernel alignment) similarity index studied by Kornblith et al [21]. This similarity index is used together with a sketching method that appears in [22], thus creating s-CKA, which was finally used in this paper. This similarity index was tested against other similarity indexes by training models to recognize 2 numbers from the MNIST dataset, and then compared the models, discovering that the s-CKA model is the most accurate means of comparing the similarity between the source models, showed in Figure 6.

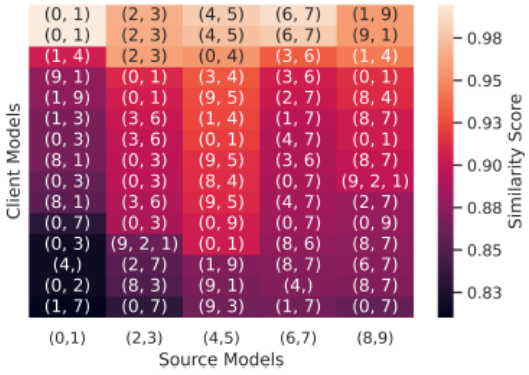

Figure 6: Test that shows the validity of the similarity score, giving results close to 1 for models recognizing the same MNIST labels. [20]

The principle of FedVote, the algorithm proposed in this paper, present in Appendix A, is very similar to the FedAvg model presented in Section 4.1.1, having a similar complexity, the only difference being that a round R_{sel} is selected in which the clients will perform a similarity check with all the source models, and voting the most similar m models. At the selected round, the most voted source model parameters will be copied to the global parameters at the server, and then the federated process continues. It seems that a proper training acceleration was achieved through the transfer process, as the number of rounds required for the algorithm to reach a 90% accuracy on a test set was improved.

Server time complexity:

$$\mathcal{O}(K \cdot M + s) \tag{9}$$

Client time complexity:

$$\mathcal{O}(train + (s \cdot (infer + sCKA))) \tag{10}$$

Message size complexity:

$$\mathcal{O}(M + K) \tag{11}$$

Symbols: M - model size; s - number of source models; sCKA - complexity of performing sCKA similarity check; infer - complexity of executing a forward inference; train - complexity of local training.

Security and Privacy
When it comes to data privacy, we can say that the data silos are safe and no data is being shared among the clients, but if a malicious user can reach the server, it can perform model inversion or reconstruction attacks by tapping into the model parameters used in the averaging process. Model inversion can also be problematic if the source models' training dataset should remain private. Moreover, data or model poisoning, which are very hard to avoid in general, can also pose a problem to this proposed framework. It can surely be concluded that this framework should only be used for its acceleration capabilities only when the parties are sure to be trustworthy.

4.1.3 Secure Federated Transfer Learning [23]

Secure Federated Transfer learning is one of the first of its type, putting the bases of privacy and security while also combining Federated Learning with Transfer Learning. In a typical application of this framework, we only have two parties, and a federation of such parties involves multiple such pairs. The aim of this framework is to assist party B, containing domain $\mathcal{D}_B = \{x_i^B\}_{i=1}^{N_B}$, into labeling its' samples x_i by using the rich labels available in $\mathcal{D}_A = \{x_j^A, y_j^A\}_{j=1}^{N_A}$.

In order to do this, the common elements of these two parties are collected, such that the samples can be labeled as $\mathcal{D}_{AB} = \{x_i^B, y_i^A\}_{i=1}^{N_{AB}}$. Then two hidden neural network representation Net^A and Net^B are created in order to project the two data distributions onto a mutual feature subspace. A loss function $\mathcal{L}_1$ is used to train the prediction function of party B, $\mathcal{L}_2$ is used for minimizing the alignment loss for feature transfer learning, and $\mathcal{L}_3$ is for the regularization terms.

Using these terms, the models perform the training phase through either homomorphic encryption and additive secret sharing, or using Beaver triples and additive secret sharing. Multiple simulations show that the computational overhead introduced by the homomorphic encryption should be considered when choosing the security technique, as Beaver triples introduces a better way to hide the model values by adding computational complexity in an unrelated offline phase in which these tuples are being generated, rather than in the training process itself.

Overall algorithm time complexity: The relevant terms that define the overall runtime complexity of the algorithm is the number of iterations, the number of common samples and the size of the models for the calculations of the losses and gradients, although using homomorphic encryption adds a serious computational overhead:

$$\mathcal{O}(R \cdot (N_{AB} \cdot (enc(M1) + enc(M2)))) \tag{12}$$

Communication size complexity: The worst-case communication size is the size of the larger model plus the size of the mask used in additive secret sharing:

$$\mathcal{O}(max(enc(M1), enc(M2)) + mask) \tag{13}$$

Symbols: R - number of rounds; N_{AB} - number co-occurrences of samples; M1 - size of the first party model; M2 - size of the second party model; mask - size of the mask used in additive secret sharing.

Security and Privacy
From a privacy perspective, the clients both manage to keep their data safe, and the models are always hiding the model parameters when sharing them using encryption or secret sharing techniques,

making targeted model attacks very hard to execute, as they can only be done in a black-box manner. Nevertheless data can still be corrupted if one of the parties is malicious and provides wrong labels and parameters in the learning process.

4.1.4 FedHealth [24]

FedHealth is a Federated Transfer Learning framework for wearable devices which aims to target the learning process in the healthcare field by training neural networks based on datasets composed of information from the clients wearing the devices on which the federated algorithm is running.

For a run of the algorithm, the server first trains a global model which is then passed to all the clients. The clients will freeze their local convolutional layers and only update the final fully connected layer, by replacing it with an alignment layer and a correlation layer, as showed in Figure 7. The model is shared to the client and back using homomorphic encryption, such that the model's parameters are being kept safe. After the model is trained on the user side, the model is being sent back to the server in an encrypted manner in order to be aligned again.

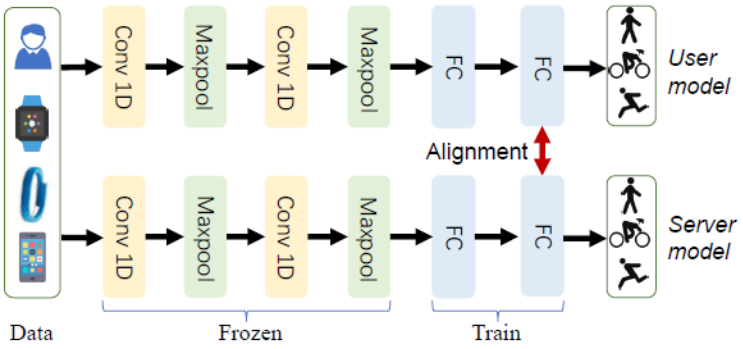

Figure 7: The alignment made on the last layer between the client and the server. [24]

The communication complexity can be kept lightweight as only the last fully connected layer can be shared, since the other layers of the models are not being changed during the alignment phase.

Overall algorithm time complexity:

$$\mathcal{O}(R \cdot (K \cdot (enc(train)))) \tag{14}$$

Message size complexity:

$$\mathcal{O}(enc(M)) \tag{15}$$

Symbols: R - number of rounds; K - number of clients; M - size of the proposed model; train - complexity of training.

Security and Privacy

Since homomorphic encryption is being used, the model parameters are being kept hidden from a malicious user. A backdoor attack can be detrimental in the scenario of FedHealth, as the first step in the algorithm is to train the source model on the server, that can corrupt the model. Since homomorphic encryption is done, only black-box access is being given to the model, so advanced reconstruction attacks are necessary. Also since differential privacy is not used, membership-inference attacks can be executed.

4.1.5 FedMD: Heterogenous Federated Learning via Model Distillation [25]

The FedMD framework provides a very intriguing view of the Federated Learning process, being an algorithm that keeps both the client data and the client models private, which is something rarely done in the federated field, as the federated algorithms are usually based on the sharing and aggregation of user models, which expose the frameworks to some possible attacks through the extraction of information from the client models. The framework deals with model heterogeneity in a novel way, allowing each client to have a different model when training the data.

The setting is defined the following way, there is a vast available dataset $\mathcal{D}_0 = (x_i^0, y_i^0)_{i=1}^{N_0}$ which can be accessed by all clients. The models f_k are free to be implemented according to the clients' wishes. The algorithm starts with the transfer phase in which the models are being trained until convergence on the publicly available dataset $\mathcal{D}_0$. After this training, for a given number of rounds P, each party will compute a score $f_k(x_i^0)$ of their model and send it to the central server which will perform an aggregation in order to reach a consensus of the form $\frac{1}{m} \cdot \Sigma_{k=1}^{m} f_k(x_i^0)$. This consensus is being sent back to the clients such that they can train their models until they approach the consensus (Figure 8), then for a certain number of epochs on their own private dataset.

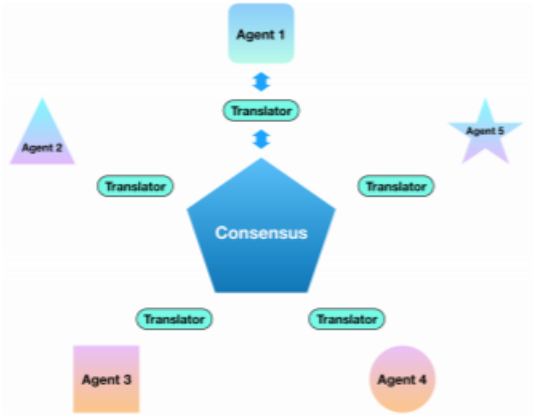

Figure 8: A general overview of the knowledge distillation mechanism, which uses the translators to coordinate the learning process by averaging them into a consensus. [25]

The communication cost of this framework is very small since the only thing that has to be sent back and forth between the client and the server is the model scoring, which can be considered constant. We can ignore the initial need to transfer the public dataset since that only has to be executed once.

Server time complexity: The server only has to calculate an average of the client scores.

$$\mathcal{O}(K) \tag{16}$$

Client time complexity: The clients train on the public dataset until convergence to average score.

$$\mathcal{O}(train) \tag{17}$$

Message size complexity: Only the score is being sent by the clients to the server

$$\mathcal{O}(1) \tag{18}$$

Symbols: K - number of clients; train - complexity of training.

Security and Privacy
Although counter-intuitive, since no security and privacy measure is being used, the algorithm is designed in such a way that the clients and the central server don't have any access to the models or datasets of the other parties, ensuring total security against attacks that seek to discover the model or the dataset of the clients. Probably the only possible attacks seek to corrupt the training samples or directly the scoring in order to change the general consensus, but if a scoring outlier detection algorithm is implemented by the central server or the number of clients is large enough, then this small corruption breach can be circumvented.

4.1.6 Federated Distillation and Augmentation under Non-IID Private Data [26]

The Federated Distillation and Augmentation framework focuses on decreasing the communication overhead of the classical Federated Learning Technique by distilling knowledge about the local models into a "logit vector". In order to guide the learning process of every model, each one considers itself as a student, and the average global "logit vector" which is calculated by the central server of the network as the teacher($\hat{F}_{k,l}^{(i)} = \Sigma_{i \neq j} \bar{F}_{k,l}^{(j)} / (\text{M-1})$). Each round a cross-entropy loss is calculated for each device between the client and teacher that aids in the training process.

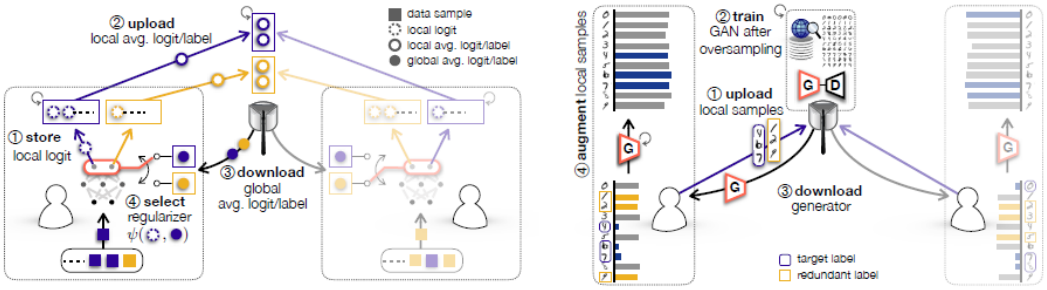

Figure 9: A visual overview of the federated distillation (left) and and augmentation (right) process. [26]

The Federated Augmentation is the second framework proposed in this paper, that allows Non Independent and Identically Distributed (henceforth shortened as non-IID) data sample distributions in a federation. It does this by training a generative adversial network (GAN) [27] prior to the learning process in order to allow all devices to fill their datasets with representative samples in order to convert the federation distribution to IID type. Every client sends samples from the labels which are too scarce, plus some redundant samples from the labels that contain enough samples in order to improve the privacy of the scarce samples. The central server uses the Google reverse image search to enrich the samples and generates the GAN that is able to generate samples for all the received labels (as seen in Figure 9).

Server time complexity: Server does the average of the logit vectors for the clients.

$$\mathcal{O}(K \cdot L \cdot logit) \tag{19}$$

Client time complexity: The client trains its model and calculates their average logit vector.

$$\mathcal{O}(E \cdot (B \cdot M) + L \cdot logit) \tag{20}$$

Message size complexity: The message complexity reduces to the size of the logit vectors passed between clients and server.

$$\mathcal{O}(logit) \tag{21}$$

Symbols: K - number of clients; M - size of the model; E - number of epochs on a client; B - size of local batch; L - number of labels; logit - size of the logit vector; train - complexity of training.

Security and Privacy
When using only the distillation algorithm in an IID federation, the privacy guarantee of the framework is fairly reliable, as reconstruction attacks are almost impossible to execute since only the final layer is being provided to the server. Federated augmentation on the other side introduces privacy leakage as samples from the clients have to be sent to the server, and the other parties will be able to generate samples for the labels that needed generative assistance. This is why redundant samples are also added, and with a more significant number of devices, the privacy leakage becomes more reasonable. Still, the paper recommends the usage of Differential Privacy at the cost of accuracy in order to prevent sensitive information from being leaked through the labels which are scarcely sampled.

4.1.7 Knowledge Federation: A Unified and Hierarchical Privacy-Preserving AI Framework [28]

Knowledge Federation (KF) proposes a conceptual hierarchical federation model that operates on multiple levels in order to achieve the goal of secure and private decentralized learning on multiple levels. It is a conceptual framework that is presented together with an implementation of the showcased techniques (the iBond platform).

The framework levels are the following:

- Information level: Information level federations are based on centralized processing of all the datasets. The computation is done directly on the data on the arbitrator, so all the data is encrypted using homomorphic encryption.
- Model level: Model level federations are classical federated learning models in which every client keeps its own data private and sends its parameters or gradients protected either by encryption, Secret Sharing or Differential Privacy to the central server for aggregation.
- Cognition level: Cognition level training is very similar to the model one, but instead of models, different embedded elements such as specific layers will be shared with the central server, which will work into fusing the local knowledge discovery in a more refined feature discovery element.
- Knowledge level: Knowledge level learning is a technique based on knowledge graphs, that allows the achievement of advanced knowledge through the use of lesser knowledge which is kept in an organized interconnected graph.

Various federations can be created at multiple levels, the Knowledge Federation platform iBond offering the user the possibility to choose between the levels. One use case is also being given when 2 different companies from the same field but with different datasets want to collaborate in a learning task regarding the calculation of credit risk score. They send their encrypted datasets to the server which finds the common elements between the two datasets. After this the important features for credit risk assessment are selected and the federated model is being built and trained. In order to keep the labels secure, the party with no labels will compute the gradients for all the labels and securely send them to the party containing the labels for aggregation.

Security and Privacy
The framework offers secure data transfer through mechanisms such as Homomorphic Encryption, Secret Sharing and Differential Privacy, making sure that the data can not be viewed by the third-party and only black box access might be given when sending models. The framework claims that the third party can also be trusted as it only acts as an arbitrator and computation assistant, not

persistently storing any data. The only possible attacks on this framework are untargeted normal data or model corruption, but the malicious user most probably does not have anything to gain from these attacks.

4.2 Framework comparison

Two aspects about the frameworks are of interest, their performance and security/privacy balance. Tables 2 and 3 present these two facets of the frameworks showcased in Section 4.1 in their order of appearance.

Algorithm	Complexity	Communication cost complexity
1.FedAvg	Server: O(K • M) Client: O(train)	O(M)
2.FedVote	Server: O(K • M + s) Client: O(train + s • (infer + sCKA))	O(M + K)
3.SecureFed Transfer	Overall: O(R • (Nab • (enc(M1) + enc(M2))))	O(max(enc(M1), enc(M2)) + mask)
4.FedHealth	O(R • (K • (enc(train))))	O(enc(M))
5.FedMD	Server: O(K) Client: O(train)	O(1)
6.FedDistil	Server: O(K • L • logit) Client: O(E • (B • M) +L • logit)	O(logit)

Table 2: Overview of the time complexities and communication cost of the frameworks presented in this Section.

When it comes to the runtime of the algorithms, we can say that the most heavyweight runtime complexities are being present in Secure Federated Transfer Learning and FedHealth as the operations are done on homomorphically encrypted data which results in heavy computational overheard. The most basic of algorithms are present in Federated Average, Vote and Model distillation, in which the training is done without any encryption. When it comes to communication cost per message, the distillation algorithms provide the best of solutions in simplifying the messaging protocol as only scores or logit vectors have to be sent over the network, which drastically increase the time efficiency of the federations using these techniques.

Algorithm	Security/Privacy Mechanism
1.FedAvg	None
2.FedVote	None
3.SecureFed Transfer	Homomorphic Encryption/Beaver Triples + Additive secret sharing
4.FedHealth	Homomorphic Encryption
5.FedMD	Model Distillation
6.FedDistil/FedAug	Knowledge Distillation
7. Knowledge Federation	SMPC/Homomorphic Encryption/Differential Privacy

Table 3: The security mechanisms that are used in the showcased frameworks.

With security in mind, various techniques are used in these frameworks, as showed in Table 3, but we can say that all of them contain at least a basic privacy-preserving mechanism since the clients participating in the federation do not share their dataset samples (with the exception of

Federated Augmentation). Federated Average and Federated Vote provide the least privacy and security and are very weak against any type of inference attack since the server has white-box access to the models, giving the opportunity for devastating attacks to an eavesdropper or someone who infiltrates the central server. The use of Differential Privacy could enhance the privacy of these two frameworks by securing them against backdoor or membership-inference attacks. Secure Federated Transfer and FedHealth provide better protection against inference attacks since only black-box access to the model is being given, giving better protection against reconstruction attacks. Besides great runtimes and communication costs, the FedMD and Federated Distillation frameworks provide the best security against inference attacks, since access to the models is not given to other parties. If a Sniper [10] algorithm is implemented on the central server of these two techniques to discard suspicious outlier scores, the frameworks would also have a great defence even against data and model poisoning from the clients. So it can be said that FedMD and Federated Distillation provide the greatest of defence in their elegant simplicity. The last conceptual framework, the Knowledge Federation, also provides all the necessary tools to prevent attacks as it is a general framework that integrates many possible defensive techniques against poisoning and inference attacks.

5 Responsible Research

The goal of research is to explore and reinforce the domain of science through either studies on the already existing elements of this domain, or through discoveries that extend the reaches of this domain. In order to make sure that research is of appropriate quality, it should be made responsibly, following well established criteria in integrity and reproducibility.

5.1 Research integrity

The first side of research integrity focuses on the veracity of the research data. In order to provide a truthful analysis of a subject, the data should not be trimmed or altered in any way, as it will impact the quality of the final result. It is better to showcase an unexpected negative result which is valid, than an invalid positive result. Thus the results of the simulation from this paper are in sync with the quality of the libraries, programming language and tools that were used to develop it, and showcase expected results after performing training on the established datasets.

The other side of integrity is focused on the authenticity of the information that has been used in the paper. Proper mention of the authors and contributors to the frameworks that have been presented in this paper is made, as the purpose of this work is exploring the current frontiers of Federated Learning combined with Transfer Learning frameworks, and how they strive to preserve privacy and security, while also offering competitive efficiency and end-result accuracy.

It is in the interest of the paper to properly present the positive and negative sides of each framework, as this work's goal is to highlight when they should be used. Without a proper and trustworthy analysis, false trust in the framework can be created leading to its possible use which can generate problems when a weak framework is presented as being very safe against all attacks. Thus the frameworks have been objectively analyzed to highlight both the advantages and disadvantages of using them.

5.2 Research reproducibility

Most of the scientific pieces that are pursuing the explanation of the inner workings of the world plan on being reproducible, in order to prove the validity of the actions and experiments undertaken in the respective studies. This is becoming an increasingly difficult tasks, as even the understanding of our reality is not sure to be deterministic [29], so at times it can be said that some experiments or actions are not completely reproducible. In such cases the directives of the setup and experiment should be really well documented, such that the results show a clear similarity to the ones presented in the study. This is a fact that should also be considered in the field of Machine Learning, as the

results of training and the model inference are highly dependent on the training set or factors such as the nature of the randomness in shuffling the data to be used in the learning process. [30] So the expected results when simulating a Machine Learning experiment should be similar to the ones in the original experiment, being extremely improbable for the results to be the same.

6 Discussion

Although Federated Transfer Learning introduces a means to utilize a scarce overlap over the sample and feature space in order to label the dataset of the party that contains only raw samples, there are situations in which the overlap is too small to achieve a positive transfer. This applies when talking about Transfer Learning in the context of feature transfer.

In cases in which pre-trained models are used in order to benefit from the knowledge already gathered by another network, the problem of negative transfer can also happen, as the task might be too vastly different for reuse. This consideration does not even include the problem of model heterogeneity, in which the models have to be moulded on the data and the features present on the target client, this problem is only solved by Model and Knowledge Distillation [25] [26], which are very straightforward and elegant solutions, but not applicable to any situation.

Another constituent problem in adding Transfer Learning to a federation is the possible need for frequent communication between the clients. Since security can introduce serious computational overhead, a Secure Federated Transfer Learning algorithm might be an inefficient solution for problems that require not only improved accuracy, but also high performance.

7 Conclusion

In situations where data privacy should be preserved at all cost, Federated Learning can be a powerful tool in decentralized training as it can enhance the quality of the entire federation of participating clients while also complying with government regulations on user privacy. The privacy of the datasets is ensured by using the federated approach, although the clients are exposed to novel network security breaches that can be solved using different security techniques, such as Homomorphic Encryption, Secure Multi-Party Computing, Differential Privacy, or hybrid and novel approaches.

For future work, the feasibility of using Transfer Learning can also be studied, as there are times in which the difference between two tasks is so great that a negative transfer will always happen. Different fields can be studied and an overview of the general fields and tasks in which Transfer Learning should be applied can be made.

As studied in this research, using a federation provides privacy and model accuracy, while various frameworks proposing different Transfer Learning mechanisms can enhance the data training process, and bring valuable contributions in cases where heterogeneity appears in the data distribution, feature representation or model design.

References

[1] S. Jurgen. "Deep learning in neural networks: An overview". In: *Neural Networks* 61 (Jan. 2015), pp. 85–117. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2014.09.003. URL: http://dx.doi.org/10.1016/j.neunet.2014.09.003.

[2] E. S. Dove. "The EU General Data Protection Regulation: Implications for International Scientific Research in the Digital Era". In: *The Journal of Law, Medicine & Ethics* 46.4 (2018), pp. 1013–1030. DOI: 10.1177/1073110518822003. eprint: https://doi.org/10.1177/1073110518822003. URL: https://doi.org/10.1177/1073110518822003.

[3] M. Brendan and R. Daniel. *Federated Learning: Collaborative Machine Learning without Centralized Training Data*. 2017. URL: https://ai.googleblog.com/2017/04/federated-learning-collaborative.html.

[4] S. Saha and T. Ahmad. *Federated Transfer Learning: concept and applications*. 2021. arXiv: 2010.15561 [cs.LG].

[5] H. B. McMahan, E. Moore, and D. Ramage. *Communication-Efficient Learning of Deep Networks from Decentralized Data*. 2017. arXiv: 1602.05629 [cs.LG].

[6] S. J. Pan and Q. Yang. "A Survey on Transfer Learning". In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (2010), pp. 1345–1359. DOI: 10.1109/TKDE.2009.191.

[7] J. S. Malhar, F. Tyler, and K. Farinaz. "A Taxonomy of Attacks on Federated Learning". In: *IEEE Security Privacy* 19.2 (2021), pp. 20–28. DOI: 10.1109/MSEC.2020.3039941.

[8] L. Yehuda. "How to Simulate It – A Tutorial on the Simulation Proof Technique". In: *Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich*. Ed. by L. Yehuda. Cham: Springer International Publishing, 2017, pp. 277–346. ISBN: 978-3-319-57048-8. DOI: 10.1007/978-3-319-57048-8_6. URL: https://doi.org/10.1007/978-3-319-57048-8_6.

[9] L. Yang Y. Qiang et al. *Federated Learning*. 2019.

[10] V. Mothukuri et al. "A survey on security and privacy of federated learning". In: *Future Generation Computer Systems* 115 (2021), pp. 619–640. ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2020.10.007. URL: https://www.sciencedirect.com/science/article/pii/S0167739X20329848.

[11] V. Feldman. "Does Learning Require Memorization? A Short Tale about a Long Tail". In: *CoRR* abs/1906.05271 (2019). arXiv: 1906.05271. URL: http://arxiv.org/abs/1906.05271.

[12] M. O. Rabin. *How To Exchange Secrets with Oblivious Transfer*. Harvard University Technical Report 81 talr@watson.ibm.com 12955 received 21 Jun 2005. 2005. URL: http://eprint.iacr.org/2005/187.

[13] D. Beaver. "Efficient Multiparty Protocols Using Circuit Randomization". In: *Advances in Cryptology — CRYPTO '91*. Ed. by F. Joan. Berlin, Heidelberg: Springer Berlin Heidelberg, 1992, pp. 420–432. ISBN: 978-3-540-46766-3.

[14] G. Craig. “Fully Homomorphic Encryption Using Ideal Lattices”. In: *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*. STOC ’09. Bethesda, MD, USA: Association for Computing Machinery, 2009, 169â178. ISBN: 9781605585062. DOI: 10.1145/1536414.1536440. URL: https://doi.org/10.1145/1536414.1536440.

[15] D. Cynthia and N. Kobbi. “Privacy-Preserving Datamining on Vertically Partitioned Databases”. In: *Advances in Cryptology – CRYPTO 2004*. Ed. by Matt Franklin. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 528–544. ISBN: 978-3-540-28628-8.

[16] M. Gharibi. *Federated Learning*. .2021. [Online]. Available: https://www.udemy.com/course/federated_learning. [Accessed: 03-May-2021].

[17] Open-Source. *PyTorch*. .2021. [Online]. Available: https://pytorch.org/. [Accessed: 02-May-2021].

[18] OpenMined. *PySyft*. .2021. [Online]. Available: https://github.com/OpenMined/PySyft. [Accessed: 05-May-2021].

[19] D. Testuggine and I. Mironov. *Introducing Opacus: A high-speed library for training PyTorch models with differential privacy*. .2020. [Online]. Available: https://ai.facebook.com/blog/introducing-opacus-a-high-speed-library-for-training-pytorch-models-with-differential-privacy/. [Accessed: 03-June-2021].

[20] T. Semwal, H. Wang, and C. K. T. Reddy. “Selective Federated Transfer Learning using Representation Similarity”. In: *NeurIPS-SpicyFL 2020 Workshop* (2020). URL: https://osf.io/kbhq5/download.

[21] S. Kornblith et al. *Similarity of Neural Network Representations Revisited*. 2019. arXiv: 1905.00414 [cs.LG].

[22] S. Tang et al. *Similarity of Neural Networks with Gradients*. 2020. arXiv: 2003.11498 [cs.LG].

[23] L. Yang et al. “A Secure Federated Transfer Learning Framework”. In: *IEEE Intelligent Systems* 35.4 (July 2020), 70â82. ISSN: 1941-1294. DOI: 10.1109/mis.2020.2988525. URL: http://dx.doi.org/10.1109/MIS.2020.2988525.

[24] Y. Chen et al. *FedHealth: A Federated Transfer Learning Framework for Wearable Healthcare*. 2021. arXiv: 1907.09173 [cs.LG].

[25] D. Li and J. Wang. *FedMD: Heterogenous Federated Learning via Model Distillation*. 2019. arXiv: 1910.03581 [cs.LG].

[26] E. Jeong et al. *Communication-Efficient On-Device Machine Learning: Federated Distillation and Augmentation under Non-IID Private Data*. 2018. arXiv: 1811.11479 [cs.LG].

[27] C. Antonia et al. “Generative Adversarial Networks: An Overview”. In: *IEEE Signal Processing Magazine* 35.1 (Jan. 2018), 53â65. ISSN: 1053-5888. DOI: 10.1109/msp.2017.2765202. URL: http://dx.doi.org/10.1109/MSP.2017.2765202.

[28] H. Li et al. *Knowledge Federation: A Unified and Hierarchical Privacy-Preserving AI Framework*. 2020. arXiv: 2002.01647 [cs.CR].

[29] V. Lev. “Quantum theory and determinism”. In: *Quantum Studies: Mathematics and Foundations* 1.1-2 (July 2014), 5â38. ISSN: 2196-5617. DOI: 10.1007/s40509-014-0008-4. URL: http://dx.doi.org/10.1007/s40509-014-0008-4.

[30] P. Henderson et al. *Deep Reinforcement Learning that Matters*. 2019. arXiv: 1709.06560 [cs.LG].

A Framework algorithms

Algorithm 1 `FederatedAveraging`. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes:
initialize w_0
for each round $t = 1, 2, \dots$ **do**
$m \leftarrow \max(C \cdot K, 1)$
$S_t \leftarrow$ (random set of m clients)
for each client $k \in S_t$ **in parallel do**
$w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
$w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

ClientUpdate(k, w): *// Run on client k*
$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
for each local epoch i from 1 to E **do**
for batch $b \in \mathcal{B}$ **do**
$w \leftarrow w - \eta \nabla \ell(w; b)$
return w to server

Figure 12: The FedAvg algorithm present in Communication-Efficient Learning of Deep Networks from Decentralized Data [5].

Algorithm 1: `FedVote` running on a client.

Input: Set of source models $\mathcal{M}_s$, global model w, R_{sel}
Output: Best m models
for *each round* $R = 1, 2, \dots$ **do**
Choose K clients
for *each client* $k \in K$ **do** `// in parallel`
$w \leftarrow$ **LocalUpdate**(k,w) `// client` k `trains model on local data`
if $R == R_{sel}$ **then**
for *each* $s \in \mathcal{M}_s$ **do**
$z_w^{l_1}, z_s^{l_2} \leftarrow w(\mathcal{P}_k), s(\mathcal{P}_k)$ `// create representations of layer` l_1 `of source model and` l_2 `of client model; local data` $\mathcal{P}_k$ `is forward passed`
$votes \leftarrow$ **s-CKA**$(z_w^{l_1}, z_s^{l_2})$ `// calculate similarity score and increment vote count for top` $m \in \mathcal{M}_s$
end
return $votes$, w to server
else
end
Server node selects the highest voted m models
end

Figure 13: The FedVote algorithm present in Selective Federated Transfer Learning using Representation Similarity [20].

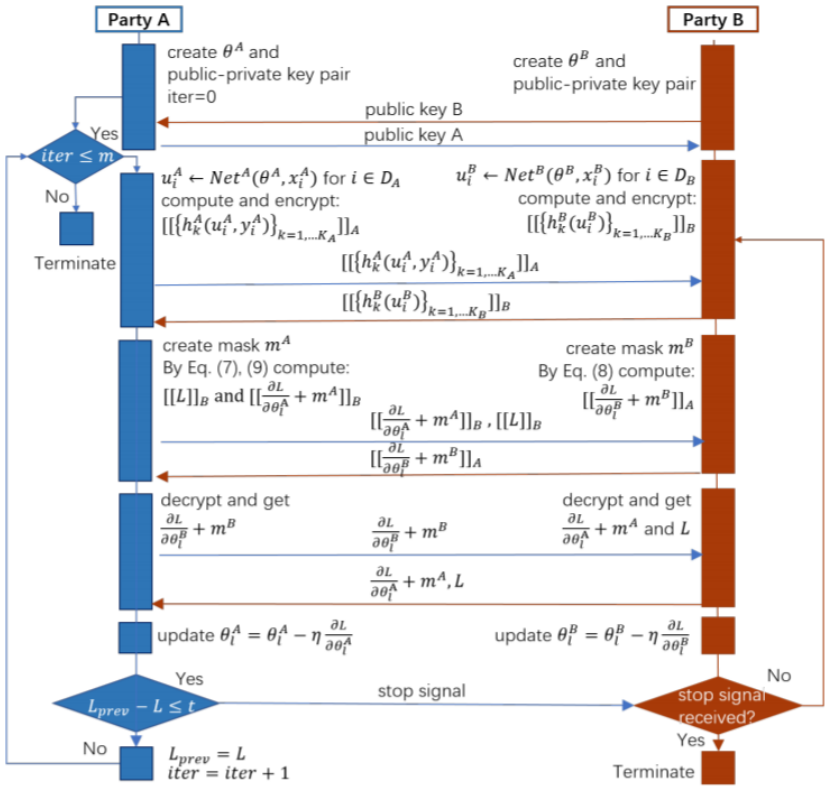

Figure 14: The HE-based algorithm scheme present in A Secure Federated Transfer Learning Framework [23].

Algorithm 1 The learning procedure of FedHealth

Input: Data from different users $\{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_N\}$, η
Output: Personalized user model f_u

1: Construct a cloud model f_S using Eq. (2)
2: Distribute f_S to all users via homomorphic encryption
3: Train user models using Eq. (3)
4: Update all user models to the server using homomorphic encryption. Then server update its model by aligning with user model
5: Distribute f_S' to all users, then perform transfer learning on each user to get their personalized model f_u using Eq. (6)
6: Repeat the above procedures with the continuously emerging user data

Figure 15: The FedHealth algorithm present in FedHealth: A Federated Transfer Learning Framework for Wearable Healthcare [24].

Algorithm 1: The FedMD framework enabling federated learning for heterogeneous models.

Input: Public dataset $\mathcal{D}_0$, private datasets $\mathcal{D}_k$, independently designed model f_k, $k = 1 \dots m$,
Output: Trained model f_k

Transfer learning: Each party trains f_k to convergence on the public $\mathcal{D}_0$ and then on its private $\mathcal{D}_k$.

for j=1,2...P **do**

Communicate: Each party computes the class scores $f_k(x_i^0)$ on the public dataset, and transmits the result to a central server.

Aggregate: The server computes an updated consensus, which is an average $\tilde{f}(x_i^0) = \frac{1}{m}\sum_k f_k(x_i^0)$.

Distribute: Each party downloads the updated consensus $\tilde{f}(x_i^0)$.

Digest: Each party trains its model f_k to approach the consensus $\tilde{f}$ on the public dataset $\mathcal{D}_0$.

Revisit: Each party trains its model f_k on its own private data for a few epochs.

end

Figure 16: The FedMD algorithm present in FedMD: Heterogenous Federated Learning via Model Distillation [25].

Algorithm 1 Federated distillation (FD)

Require: Prediction function: $F(w, input)$, Loss function: $\phi(F, label)$, Ground-truth label: y_{input}

while not converged **do**
  procedure LOCAL TRAINING PHASE (at each device)
    for n steps **do** : $B, y_B \leftarrow \mathbb{S}$
      for sample $b \in B$ **do**
        $w^{(i)} \leftarrow w^{(i)} - \eta\nabla\{\phi(F(w^{(i)}, b), y_b) + \gamma \cdot \phi(F(w^{(i)}, b), \hat{F}^{(i)}_{k,y_b})\}$
        $F^{(i)}_{k,y_b} \leftarrow F^{(i)}_{k,y_b} + F(w^{(i)}, b)$, $cnt^{(i)}_{k,y_b} \leftarrow cnt^{(i)}_{k,y_b} + 1$
    for label $\ell = 1, 2, \cdots, L$ **do**
      $\bar{F}^{(i)}_{k,\ell} \leftarrow F^{(i)}_{k,\ell}/cnt^{(i)}_{k,\ell}$: **return** $\bar{F}^{(i)}_{k,\ell}$ to server
  procedure GLOBAL ENSEMBLING PHASE (at the server)
    for each device $i = 1, 2, \cdots, M$ **do**
      for label $\ell = 1, 2, \cdots, L$ **do**
        $\bar{F}_{k,\ell} \leftarrow \bar{F}_{k,\ell} + \bar{F}^{(i)}_{k,\ell}$
    for each device $i = 1, 2, \cdots, M$ **do**
      for label $\ell = 1, 2, \cdots, L$ **do**
        $\hat{F}^{(i)}_{k+1,\ell} \leftarrow \bar{F}_{k,\ell} - \bar{F}^{(i)}_{k,\ell}$, $\hat{F}^{(i)}_{k+1,\ell} \leftarrow \hat{F}^{(i)}_{k+1,\ell}/(M-1)$: **return** $\hat{F}^{(i)}_{k+1,\ell}$ to device i
end while

Figure 17: The Federated Distillation algorithm present in Federated Distillation and Augmentation under Non-IID Private Data [26].