Problems of Privacy and Heterogeneity for Federated Learning Applications in Medical Image Analysis

Aliaksei Himbitski
Belarusian State University
Minsk, Belarus
Email: alekseygimbickiy@gmail.com

Victor Zelenkovsky
Belarusian State University
Minsk, Belarus

Maksim Zhydovich
Belarusian State University
Minsk, Belarus

Vassili Kovalev
The United Institute of Informatics Problems of the National Academy of Sciences of Belarus biomedical image analysis laboratory
Minsk, Belarus

***Abstract*—Recently, machine learning has become one of the most promising directions in working with medical data. Deep neural network models are the most effective and accurate, but they require large volumes of information for training. This is a common problem in the case of medical data, especially images, as their creation involves significant costs.**

One solution to improve the quality of deep neural network models without increasing the training dataset is model aggregation. However, a problem arises with preserving the confidentiality of medical images. For example, if one model is trained on an image containing information about a specific patient, other models participating in the aggregation may also gain access to this information. As a result, information about a specific patient may be disclosed.

In an attempt to address the problem described above, this work aims to research and develop methods for aggregating machine learning models while preserving the privacy of medical images, particularly federated learning methods.

***Keywords*—Computer vision, machine learning, deep neural network, medical images analysis, image processing**

I. Introduction

In the modern world, deep neural networks are one of the most powerful tools for analyzing medical data, as they can extract complex relationships between different features. However, one of the main challenges in training large and very large neural networks is the computational limitations during training. This is due to the fact that training deep neural networks involves using backpropagation algorithms, which require a large number of iterations to achieve good training quality. And as the size of the neural network increases, the number of iterations required for training increases exponentially.

Thus, training large neural networks becomes highly challenging and demands significant computational resources. One way to address this problem is through model aggregation. Model aggregation in machine learning is the process of combining multiple models into a more powerful and efficient one.

II. Idea of federated learning

The idea of federated learning emerged and was first described in 2016 by researchers from Google in their paper titled "Communication-Efficient Learning of Deep Networks from Decentralized Data" [1]. This approach belongs to the domain of distributed machine learning, deployed across multiple clients, and enables training a unified global model on the server using several sources (clients), each of which is trained on its own dataset. More formally, let there be N clients $C1, C2, ..., C_N$ participating in the construction of the global statistical model, each with its own dataset $D1, D2, ..., D_N$. The server coordinates the work of different clients and their training

The process of federated learning can be divided into three key stages:

1) Initialization: At each step t, clients download the latest version of the model wt from the server.
2) Local training: Each client C_k performs iterative training based on its own local dataset D_k and a hyperparameter η. The client updates the weights of its local model after several training epochs, denoted as $w_t^k \leftarrow w_{t-1}^k(\eta, D_k)$, and sends them back to the server.
3) Model aggregation: The server performs the aggregation of weights received from the local models and updates the global model

$$\omega_t^{global} \leftarrow Agregation(w_t^k, k \in 1, 2, ..., N) \qquad (1)$$

The goal of the entire process is to minimize the objective function, which can be expressed by the following formula:

$$min_w \sum_{k=1}^{N} p_k F_k(w), \tag{2}$$

where F_k is the local objective function for the k-th client, pk is a value reflecting the relative influence of each client, with $p_k > 0$ and $\sum_{k=1}^{N} p_k = 1$. In other words, at each step, each client updates the weights of its model, and then the server aggregates k sets of weights using a specific aggregation method (such as averaging aggregation, progressive Fourier aggregation, FedGKT, etc.). More detailed, the entire process of federated learning is shown in the figure 1.

Later, this strategy was referred to as centralized federated learning, and a decentralized federated learning strategy was also proposed. In the decentralized federated learning strategy, there is no need for a central server with which the model clients exchange data. Instead, each client individually communicates with some other clients and aggregates their updates. This strategy helps address the issue of a single point of failure, where the entire model does not break down due to isolated errors.

The main advantages of federated learning can be summarized as follows [2]:

- Scalability: The distributed nature of federated learning allows the system to easily adapt to changes
- Model simplification: By enabling different collaborating devices to conduct multiple parallel training cycles with small amounts of data, federated learning simplifies the traditional centralized approach, where a single entity has to process a substantial volume of data each time. in the number of participating devices.
- Faster convergence: By using simpler models, devices participating in federated learning can perform multiple iterations more quickly since they learn from the experiences of other devices, leading to the faster development of a reliable global model.

However, despite the aforementioned advantages, federated learning methods have limitations when it comes to the heterogeneity of data and local models. This article focuses on the analysis and solution of these specific challenges.

III. Federated Learning Algorithms

A. Federated Stochastic gradient descent (FedSGD)

In a typical machine learning system, an optimization algorithm like stochastic gradient descent (SGD) works with a large dataset uniformly distributed across servers in the cloud. The gradient is computed on a mini-batch, which is a random subset of the original data, for each step. However, in the case of federated learning, the data is distributed unevenly across millions of devices, and some devices may be unavailable at certain times.

To address these challenges, a modification of SGD called Federated SGD has been introduced. In this approach, the gradients are averaged by the server in proportion to the number of training samples on each node and used for performing the gradient descent step [3].

$$F_k(w) = \frac{1}{n_k} \sum_{i \in P_k} (w) \tag{3}$$

$$g_k = \nabla F_k(w_t) \tag{4}$$

$$w_{t+1} \leftarrow w_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} g_k \tag{5}$$

B. Federated averaging (FedAvg)

Federated averaging is a generalization of FedSGD that allows local nodes to perform more than one batch update on their local data before exchanging weights with other models. Now, instead of exchanging gradients, local models exchange weights. The rationale behind this generalization is that if all local models start from the same initialization, averaging gradients is strictly equivalent to averaging the weights themselves.

$$\forall k, w_{t+1}^k \leftarrow w_t - \eta g_k \tag{6}$$

$$w_t + 1 \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k \tag{7}$$

C. Federated learning with dynamic regularization (FedDyn)

In cases where the data from different client models is not homogeneous, minimizing the local loss functions of the clients may not decrease the global loss function. Therefore, it has been proposed to use regularization with the use of data statistics, such as data volume, transmission speed, and cost. This modification transforms the values of local loss functions into values of global loss functions.

D. FedProx

This method is an improvement over the previous approach and aims to mitigate the issue of local optimization inherent in stochastic gradient descent-based methods.

The problem at hand is as follows: performing numerous local iterative training steps in FedAvg can cause each client to prioritize achieving its own local objective rather than the global one, leading to suboptimal convergence or model divergence.

The solution proposed by this method involves adding a term $\frac{\mu}{2}\|w_k^t - w_{global}^t\|^2$ to the objective function to regulate the influence of local models and ensure convergence guarantees. When $\mu = 0$, FedProx is equivalent to FedAvg, meaning that subsequent model aggregation and global updates are performed using the same principles as in FedAvg.

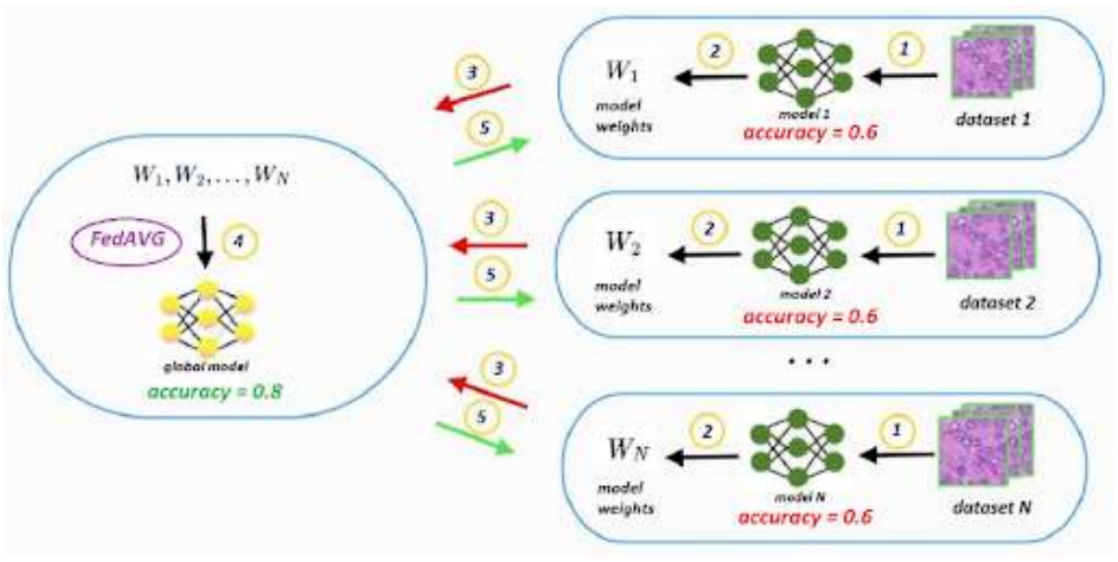

Figure 1: Diagram of the federated learning process

E.FedNova

In the FedNova algorithm, the model aggregation stage of the FedAvg algorithm is modified to address the issue of model non-identicality. Before updating the global model, the algorithm applies normalization and scaling to the local updates from each client based on their local iteration number.

IV. The Problem of Heterogeneity in Federated Learning

Existing methods of federated learning have the following drawbacks that limit their use for medical imaging:

Data privacy concerns: During the aggregation phase, the server receives all model weights from the clients, which leads to a loss of data confidentiality. By having access to all weights and information about the model's hyperparameters, it becomes possible to reconstruct the images on which the local models were trained with some level of accuracy.

Difficulty in aggregating models with different architectures: In cases where the architectures of the local models differ significantly, the number and dimensions of weight matrices also differ greatly. This makes it impossible to apply basic federated learning methods.

Difficulty in aggregating heterogeneous data: Often, the data on clients may undergo different preprocessing. Moreover, the data may differ in class imbalance, which further complicates the use of basic federated learning methods.

To address these issues, a method is proposed that distinguishes itself from other approaches by having the server receive only a specific portion of the model weights during the aggregation phase. This feature allows for the preservation of the privacy of the local model and, consequently, the data on which it was trained.

Weight aggregation still occurs on the server, but now the server itself is a machine learning model (specifically, a neural network) trained on a similar task and data that is similar to the data used to train the local models.

1) Clients send a specific portion (not all) of the trainable weights to the server.
   $w_k^t, k \in [1, 2, ..., N]$
2) The server uses a transformation $F : R_k \rightarrow R_h$ to convert the received weights w_k^t from the R_k space (the weight space of the k-th client, where the weights transmitted by the clients may have different dimensions) to the hidden space R_h. This is done for the convenience of aggregating the weights, which are now in a unified space.

$$w_k^t \leftarrow F(w_k^t) \tag{8}$$

3) Next, using the transformation $G : R_h - > R_h$, the server aggregates the weights obtained in the previous step as follows:

$$w_{global} \leftarrow G(w_k^t) \tag{9}$$

4)) Then, for all clients, the server applies the inverse transformation $F^{-1} : R_h - > R_k$ to convert the aggreted weights w_{gloval} back to the original weight space of the client R_k and sends them back to the client.

The described method has the same advantages as basic federated learning approaches, but it has an additional