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 C_1, C_2, ..., C_N participating in the construction of the global statistical model, each with its own dataset D_1, D_2, ..., 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 $w_t^k \leftarrow w_{t-1}^k$ (η, 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.

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

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 in the number of participating devices.
- 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.
- 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].

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

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

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

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 μ = 0, FedProx is equivalent to FedAvg, meaning that subsequent model aggregation and global updates are performed using the same principles as in FedAvg.

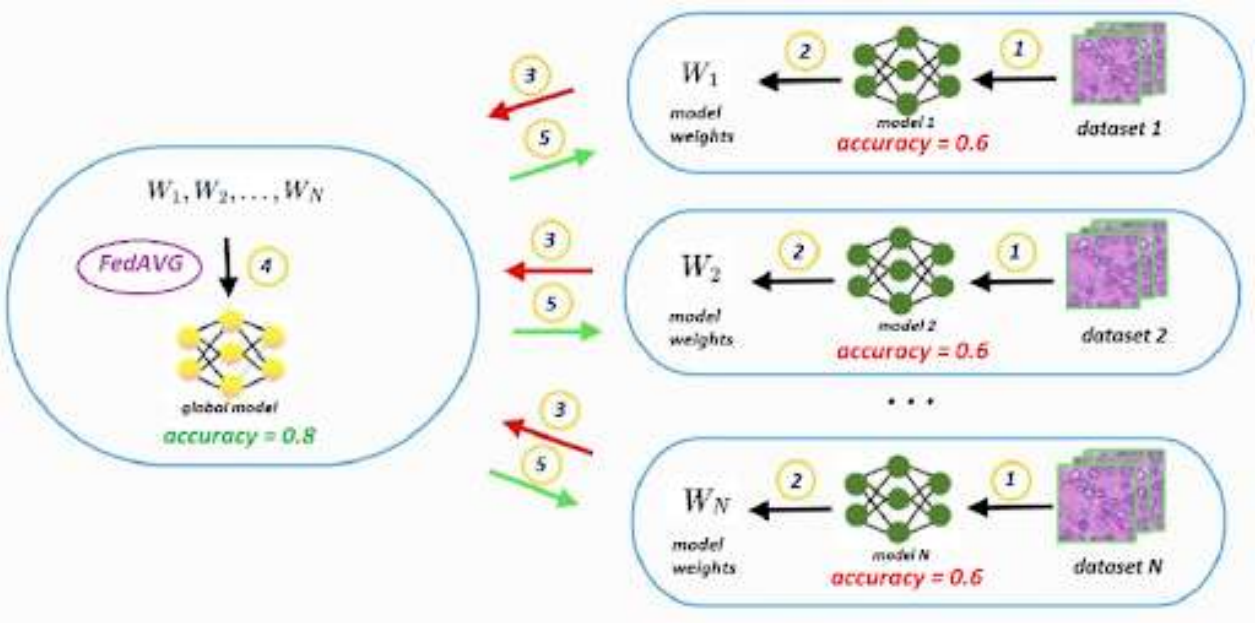

Figure 1. Diagram of the federated learning process.

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 aggregated weights w_{global} 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

key advantage that other methods lacked: preserving the privacy of images.

Data leakage does not occur because clients only exchange a portion of the weights with the server, which is insufficient to reconstruct the original data on which the model was trained.

It is precisely this advantage that enables the use of this algorithm for training models used for medical images, ensuring the privacy and confidentiality of sensitive patient data.

Among the drawbacks of this approach, the following can be identified:

- The need for additional training of the server model: This requires extra time and additional data for training the server model.
- Lack of improvement in model quality when the number of weights transmitted by clients for aggregation is too small: If the amount of weight information shared by clients is insufficient, the overall model quality may not improve significantly.
- Potential degradation in model quality when the architectures of the models differ significantly: If the models used by the clients have vastly different architectures, the aggregation process may lead to a decrease in model quality instead of improvement.

V. Experiments and Results

For the analysis of the effectiveness of the developed method, a dataset of 12,000 histological images was used, divided into two classes: malignant tumors and benign tumors. The dataset consisted of 8,400 images for training and 3,600 images for testing. The training dataset was randomly divided into five parts: four clients and one server.

The training process involved a cycle of weight exchange between the clients and the server, followed by aggregation and sending back of the weights. This cycle was performed every 3 epochs of training, and a total of 10 cycles were conducted. The weights for aggregation were sent from each client. Therefore, the total number of training epochs for each client was 30.

The weights of the models for exchange (R_k space) were the weight matrices of the linear classification layer of the network, with a dimension of 1024x1024 for all clients.

The evaluation of the method was done by comparing the following values obtained with and without the application of this method during training (traditional training without weight aggregation for 30 epochs):

- The meaning of the loss function (cross-entropy) during training.
- The value of the loss function on the test dataset.
- The accuracy of predictions on the test dataset.

A. Models with the same architecture

For analyzing the effectiveness of the developed method in the case of homogeneity of local models, was used a simple neural network with the architecture shown in the figure 2:

```
       Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 16, 222, 222]             448
         MaxPool2d-2           [-1, 16, 74, 74]               0
            Conv2d-3           [-1, 16, 72, 72]           2,320
            Conv2d-4           [-1, 32, 70, 70]           4,640
         MaxPool2d-5           [-1, 32, 23, 23]               0
           Flatten-6                [-1, 16928]               0
            Linear-7                 [-1, 1024]      17,335,296
            Linear-8                 [-1, 1024]       1,049,600
            Linear-9                    [-1, 2]           2,050
================================================================
Total params: 18,394,354
Trainable params: 18,394,354
Non-trainable params: 0
```

Figure 2. Simple neural network architecture.

As the global model (server), a pre-trained resnet18 network was used, fine-tuned on 1170 histological images for 30 epochs. For weight aggregation from clients, the last layer designed for classification was modified to the following:

For weight aggregation, a linear layer called "aggregate" is used. It takes weights from clients as input for aggregation and produces modified weights for each client as output.

In the figure 4 is a graph showing the evaluation of various quality metrics for client 0 (the graphs for other clients are similar).

Based on the analyzed data graphs, the following conclusions can be drawn about the performance of this method on simple models of the same architecture:

- For all clients, there is a decrease and stability in the values of the loss function during training when the method is applied.
- For 3/4 of the clients, there is higher stability in the accuracy of predictions on the test dataset when the method is applied.
- The average prediction accuracy did not change when the method was applied.

B. Models of various architectures

The following pretrained neural networks were used to analyze the effectiveness in the case of heterogeneity of local models:

- Client 1: SimpleModel (see above);
- Client 2: MobileNetV3 Large;
- Client 3: MobileNetV3 Small;
- Client 4: DenseNet121.

For each local model, the last layer of the neural network was replaced with the layer shown in the figure 5:

The transferred weights are the weights of the shared linear layer. The model for the server is similar to the model from the previous section.

```
(fc): Sequential(
  (fc): Linear(in_features=512, out_features=1048576, bias=True)
  (reshape): Reshape()
  (aggregate): Sequential(
    (0): Linear(in_features=1024, out_features=1024, bias=True)
    (1): ReLU()
  )
  (max_pool): MaxPool2d(kernel_size=(1024, 1), stride=(1024, 1), padding=0, dilation=1, ceil_mode=False)
  (squeeze): Squeeze()
  (classifier): Linear(in_features=1024, out_features=2, bias=True)
)
```

Figure 3. The last layer of the resnet18 network for weight aggregation.

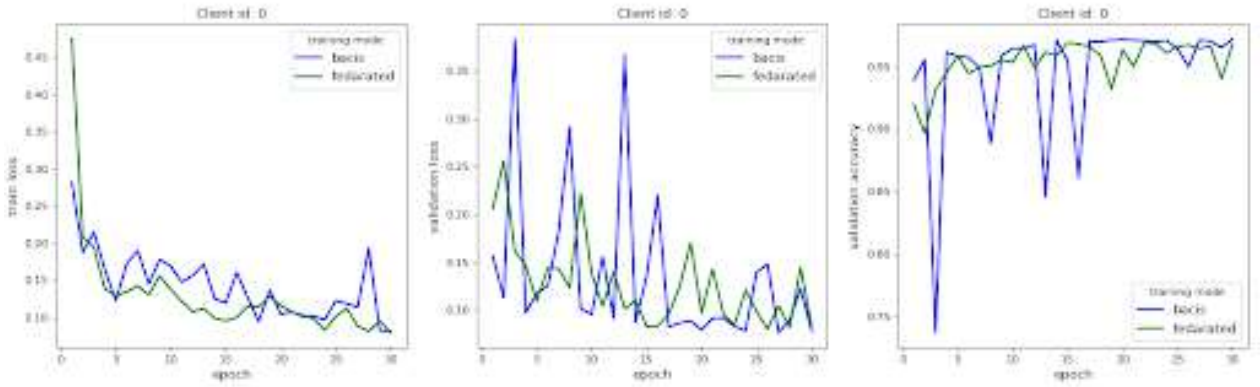

Figure 4. Graphs of analyzed metrics (SimpleModel).

```
(classifier): Sequential(
  (fc): Linear(in_features=1024, out_features=1024, bias=True)
  (relu1): ReLU()
  (shared): Linear(in_features=1024, out_features=1024, bias=True)
  (relu2): ReLU()
  (classifier): Linear(in_features=1024, out_features=2, bias=True)
)
```

Figure 5. The last layer of the local models.

In figures 6 and 7 are the plots showing the evaluation of various quality metrics for clients 0 and 2.

Based on the analyzed data graphs, the following conclusions can be drawn about the performance of this method on pre-trained models of various architectures:

- For all clients, there is a decrease and stability in the values of the loss function during training when the method is applied.
- There is higher stability in the prediction accuracy on the test dataset for simpler models and some pre-trained networks when the method is applied.
- On average, the prediction accuracy with the method applied has not changed.

C. Heterogeneous data

In real life, it is possible for data to be stored on different media. Additionally, the data can be heterogeneous.

To analyze the effectiveness in the case of data heterogeneity, consider a scenario where the original dataset is divided into two clients. The data of the first client contains 95% of class 0 objects and 5% of class 1 objects. Conversely, the second client has 5% of class 0 objects and 95% of class 1 objects. Architecture of client models is shown below

As the global model a pre-trained resnet18 network was used. For weight aggregation from clients, the last layer designed for classification was modified to the layer in the figure 9

During the training process of the second and third models, federated learning was used. The FedAVG algorithm was selected for aggregating information from client models.

The exchange of model weights between the client models and the server occurred every 3 epochs. Each of the client models was trained for a total of 30 epochs. The cross-entropy loss function was used.

The goal of the experiment was to study the dependence of model prediction accuracy on the partitioning of the dataset among clients.

Thus, we simulate a situation where the data from different clients is highly heterogeneous

In such a situation, it is not possible to achieve

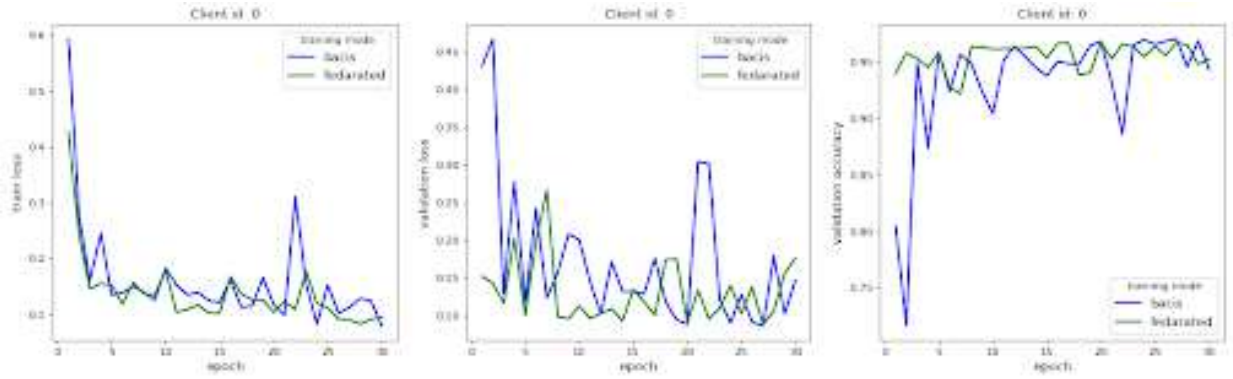

Figure 6. Graphs of analyzed metrics (SimpleModel).

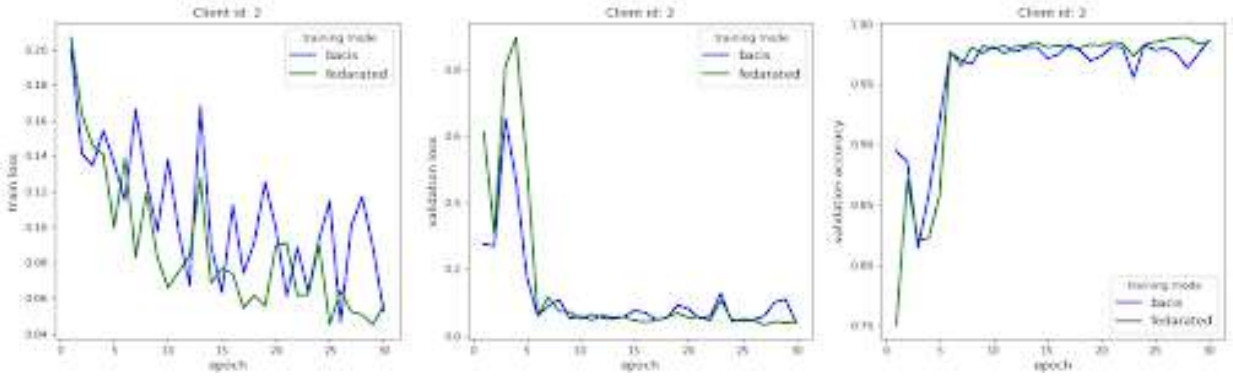

Figure 7. Graphs of analyzed metrics (MobileNetV3 Large).

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 16, 222, 222]	448
MaxPool2d-2	[-1, 16, 74, 74]	0
Conv2d-3	[-1, 16, 72, 72]	2,320
Conv2d-4	[-1, 32, 70, 70]	4,640
MaxPool2d-5	[-1, 32, 23, 23]	0
Flatten-6	[-1, 16928]	0
Linear-7	[-1, 1024]	17,335,296
Linear-8	[-1, 1024]	1,049,600
Linear-9	[-1, 2]	2,050

Total params: 18,394,354
Trainable params: 18,394,354
Non-trainable params: 0

Figure 8. Client model architecture.

```
(fc): Sequential(
  (fc): Linear(in_features=512, out_features=1048576, bias=True)
  (reshape): Reshape()
  (aggregate): Sequential(
    (0): Linear(in_features=1024, out_features=1024, bias=True)
    (1): ReLU()
  )
  (max_pool): MaxPool2d(kernel_size=(1024, 1), stride=(1024, 1), padding=0, dilation=1, ceil_mode=False)
  (squeeze): Squeeze()
  (classifier): Linear(in_features=1024, out_features=2, bias=True)
)
```

Figure 9. The last layer of the resnet18 network for weight aggregation.

stable learning and good results when using sequential learning, where one client is trained first and then the other. However, by using federated learning methods, we were able to achieve the same metric values as in the case of balanced data. This is primarily because, during aggregation, the global model aims to acquire knowledge from all clients and average them.

Each of the client models learned to predict the classes present in its own dataset very well. However, when it comes to predicting classes that were rarely encountered before, the client models struggle on the test data. Nonetheless, the central model aggregated information from all the data, which can potentially improve its ability to predict such classes.

Thus, federated learning methods enable us to mitigate the impact of data heterogeneity when training models.

In the figure 10 is shown the graph of accuracy for heterogeneous data.

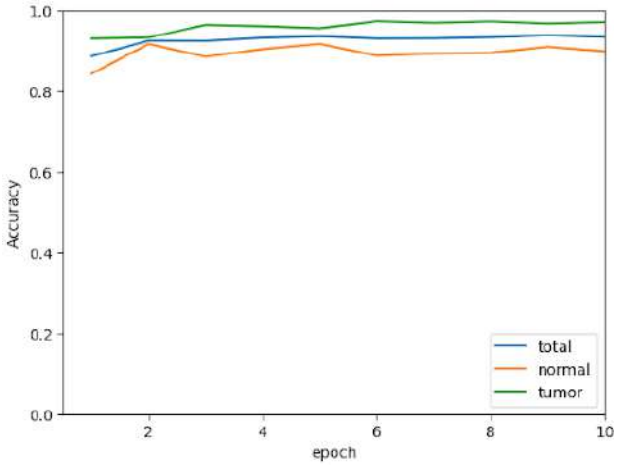

Figure 10. Graph of accuracy for heterogeneous data.

VI. Semantic technologies application

In addition to the above applications, you can use semantic technologies in federated learning. For example as follows.

- Distributed knowledge representation: Semantic technologies enable the representation of knowledge in a structured manner using formal languages. This can be useful in distributed model training, where each device can have its local knowledge representation and then combine these representations on a central server.
- Semantic data analysis: Semantic technologies can assist in the analysis and understanding of data collected from distributed devices. For example, they can be used to extract meaning and relationships between data, which can be beneficial for aggregation and merging of models on a central server.
- Unification of semantic understanding: Semantic technologies can help unify the understanding of data across different devices or servers. They can be used to create a shared model of knowledge or a semantic network, which can be utilized for harmonizing and collaboratively training models on different devices.

VII. Conclusion

In this article, the main problems arising in the task of biomedical image analysis were described, and a method to avoid them was proposed. Furthermore, experiments were conducted to demonstrate its practical applicability. Has been shown that federated learning helps preserve data confidentiality and also provides a significant improvement in quality when different clients have data from different classes. Various approaches to solving the problem of heterogeneous learning have been considered. The obtained conclusions and recommendations can be valuable for researchers in the field of biomedical informatics and medicine who aim to utilize advanced machine learning methods for image analysis in privacy-preserving conditions.

Acknowledgment

This research was supported by the United Institute of Informatics Problems of the National Academy of Sciences of Belarus (UIIP NASB).

References

[1] H. Brendan McMahan, E. Moore, D. Ramage, S.Hampson, B. Agüera y Arcas, “Communication-efficient learning of deep networks from decentralized data,” Artificial Intelligence and Statistics, 1273–1282 (2017).

[2] P. Qi, D. Chiaro, A. Guzzo, M. Ianni, G. Fortino, F. Piccialli “Model aggregation techniques in federated learning: A comprehensive survey”. Future Generation Computer System, 150, 272-293 (2023).

[3] Federated Learning for Mobile Keyboard Prediction, McMahan, Brendan, et al., 2017.

[4] EU. Regulation (eu) 2016/679 of the european parliament and of the council on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation). https://eur-lex.europa.eu/legal-content/EN/TXT., 2018.

[5] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communicationefficient learning of deep networks from decentralized data. In Aarti Singh and Xiaojin (Jerry) Zhu, editors, Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA, volume 54 of Proceedings of Machine Learning Research, pp. 1273–1282. PMLR, 2017.

[6] Kairouz Peter, McMahan H Brendan, Avent Brendan, Bellet Aurélien, Bennis Mehdi, Bhagoji Arjun Nitin, Bonawitz Keith, Charles Zachary, Cormode Graham, Cummings Rachel, et al. Advances and open problems in federated learning. arXiv preprint arXiv:1912.04977, 2019.

[7] Hard Andrew, Rao Kanishka, Mathews Rajiv, Ramaswamy Swaroop, Beaufays Françoise, Augenstein Sean, Eichner Hubert, Kiddon Chloé, and Ramage Daniel. Federated learning for mobile keyboard prediction. arXiv preprint arXiv:1811.03604, 2018.

[8] Jin Yilun, Wei Xiguang, Liu Yang, and Yang Qiang. A survey towards federated semi-supervised learning. arXiv preprint arXiv:2002.11545, 2020.
[9] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. IEEE Signal Process. Mag., 37(3):50–60, 2020.
[10] Zengpeng Li, Vishal Sharma, and Saraju P. Mohanty. Preserving data privacy via federated learning: Challenges and solutions. IEEE Consumer Electronics Magazine, 9(3):8–16, 2020.

ПРОБЛЕМЫ КОНФИДЕНЦИАЛЬНОСТИ И НЕОДНОРОДНОСТИ ПРИЛОЖЕНИЙ ФЕДЕРАТИВНОГО ОБУЧЕНИЯ ПРИ АНАЛИЗЕ МЕДИЦИНСКИХ ИЗОБРАЖЕНИЙ

Гимбицкий А. В., Зеленковский В. П.,
Жидович М. С., Ковалёв В. А.

В последнее время машинное обучение стало одним из самых многообещающих направлений в работе с медицинскими данными. Модели глубоких нейронных сетей являются наиболее эффективными и точными, но требуют больших объемов информации для обучения. Это общая проблема в случае медицинских данных, особенно изображений, так как их создание включает значительные затраты. Одним из решений для повышения качества моделей глубокого обучения без увеличения обучающего набора данных является агрегация моделей. Однако возникает проблема сохранения конфиденциальности медицинских изображений. Например, если одна модель обучается на изображении, содержащем информацию о конкретном пациенте, другие модели, участвующие в агрегации, также могут получить доступ к этой информации. В результате информация о конкретном пациенте может быть раскрыта.

В попытке решить описанную выше проблему, данная работа направлена на исследование и разработку методов агрегации моделей машинного обучения с сохранением конфиденциальности медицинских изображений, в особенности методов федеративного обучения.

Received 13.03.2024