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.

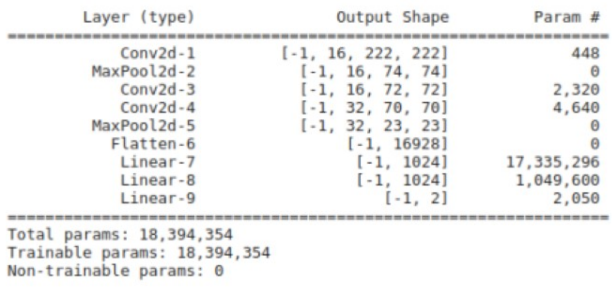
The weights for aggregation were sent from each client. Therefore, the total number of training epochs for each client was 30. The weights of the models for exchange (Rk space) were the weight matrices of the linear classification layer of the network, with a dimension of 1024x1024 for all clients.

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 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 1: 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.

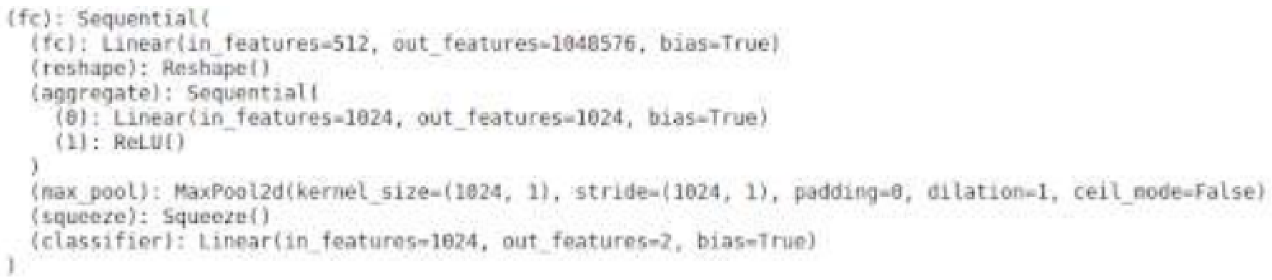
B. Models of various architectures

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 2: The last layer of the resnet18 network for weight aggregation.

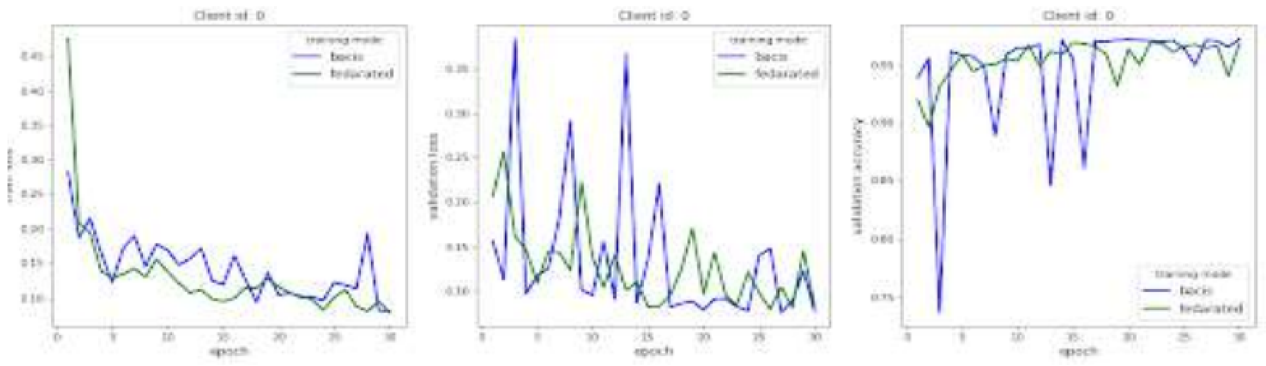
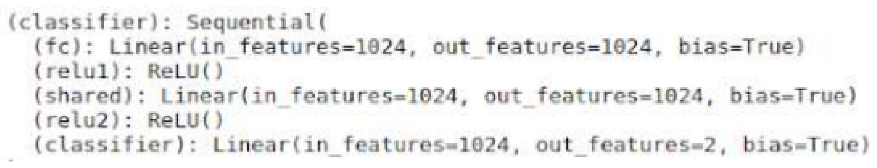


Figure 3: Graphs of analyzed metrics (SimpleModel).



```
(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 4: 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 pretrained 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 5Conversely, the second client has 5and 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

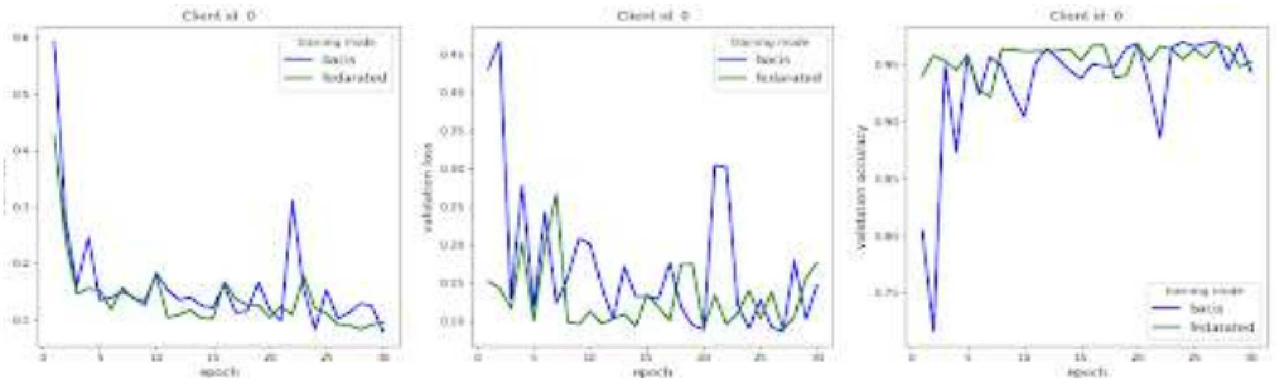


Figure 5: Graphs of analyzed metrics (SimpleModel).

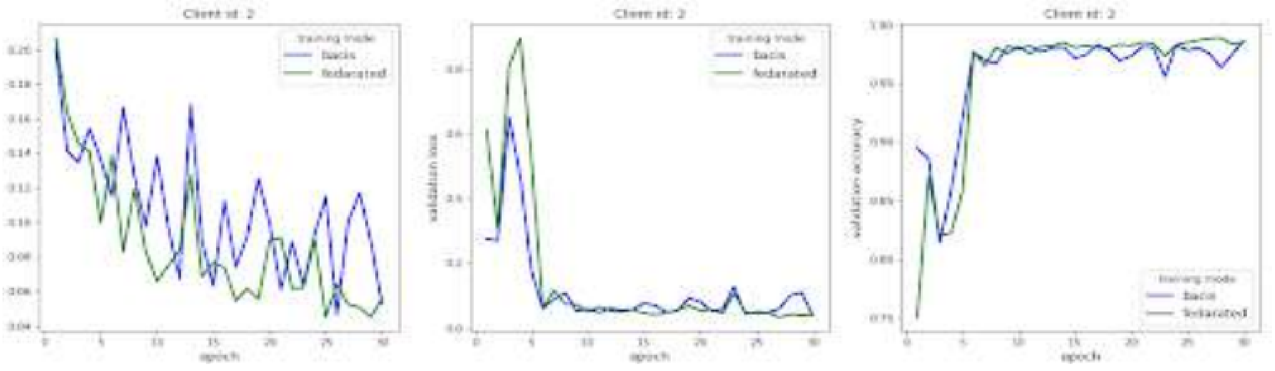
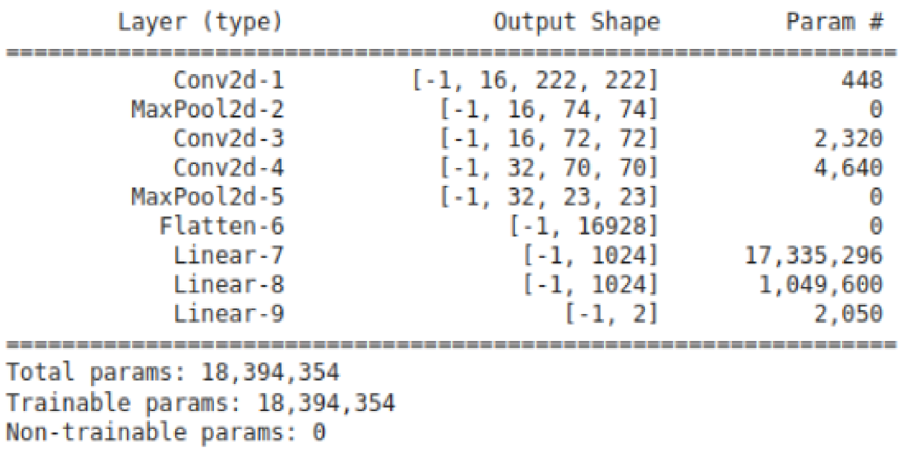
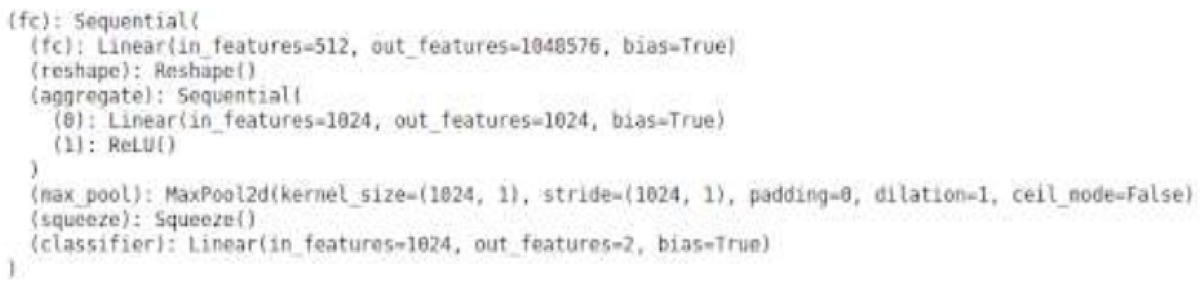


Figure 6: Graphs of analyzed metrics (MobileNetV3 Large).



```
        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 7: 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 8: The last layer of the resnet18 network for weight aggregation.