PAPER

Federated Learning of Neural ODE Models with Different Iteration Counts

Yuto HOSHINO[†a)], **Hiroki KAWAKAMI**[†b)], ***Nonmembers***, ***and*** **Hiroki MATSUTANI**[†c)], ***Member***

SUMMARY Federated learning is a distributed machine learning approach in which clients train models locally with their own data and upload them to a server so that their trained results are shared between them without uploading raw data to the server. There are some challenges in federated learning, such as communication size reduction and client heterogeneity. The former can mitigate the communication overheads, and the latter can allow the clients to choose proper models depending on their available compute resources. To address these challenges, in this paper, we utilize Neural ODE based models for federated learning. The proposed flexible federated learning approach can reduce the communication size while aggregating models with different iteration counts or depths. Our contribution is that we experimentally demonstrate that the proposed federated learning can aggregate models with different iteration counts or depths. It is compared with a different federated learning approach in terms of the accuracy. Furthermore, we show that our approach can reduce communication size by up to 89.4% compared with a baseline ResNet model using CIFAR-10 dataset.

key words: *federated learning, neural networks, neural ODE*

1. Introduction

In traditional cloud-based machine learning systems, sending personal data to cloud servers has become problematic from a privacy perspective. Federated learning [1] is a distributed machine learning approach that can keep privacy-sensitive raw data decentralized. In the federated learning, clients receive a model from the server. Then they train the model with their own data and upload trained parameters to the server. The server aggregates the trained parameters received from the clients and sends back the aggregated parameters to the clients. These steps are repeated until the training process is converged. This eliminates the need to upload privacy-sensitive raw data to the server.

However, there are some challenges in the federated learning, such as communication size reduction and client heterogeneity. Communication size affects communication delay and power consumption of clients. It is affected by the machine learning model size. Regarding the client heterogeneity, not all clients always have the same hardware, compute resources, or training data. A client may use a deeper model for high accuracy, while another client may use a shallower model to reduce the computation cost. In this paper, we exploit Neural ODE [2] as a federated learning model to address these challenges.

For image recognition tasks, one of methods to improve accuracy is increasing the number of convolutional layers to build a deeper neural network. ResNet [3] is one of well-known CNN models that stack many residual blocks that contain convolutional layers and shortcut connections. Neural ODE utilizes a similarity to ODE (Ordinary Differential Equation) to implement deep neural networks consisting of residual blocks. Since it can be approximated to a ResNet model by repeatedly using the same weight parameters, it can reduce the weight parameters. In addition, it can be approximated to ResNet models with different depths by changing the iteration counts without increasing the number of parameters. dsODENet [4] is a lightweight model that combines the ideas of Neural ODE and depthwise separable convolution [5] to further reduce the parameter size and computation cost. These Neural ODE models are smaller than ResNet.

In this paper, we introduce a flexible federated learning that allows clients to use models with different iteration counts and reduces the communication size by using Neural ODE based models. Our contributions are listed below*.

- We propose to use the Neural ODE models for federated learning so that a server can aggregate models with different iteration counts. This can enhance the client heterogeneity since clients can use models with different iteration counts. In addition, using the Neural ODE models can significantly reduce the communication size.
- We experimentally demonstrate that the proposed flexible federated learning can aggregate these models with different iteration counts. It is compared with a federated learning approach that uses knowledge distillation. We discuss the pros and cons of the proposed approach.

The rest of this paper is organized as follows. Section 2 introduces baseline technologies behind our proposal. Section 3 proposes the flexible federated learning approach and shows the feasibility of the proposed approach. Section 4 evaluates the proposed approach in terms of the accuracy and communication size. Section 5 discusses pros and cons of the proposed approach against counterparts. Additional evalu-

Manuscript received September 2, 2023.
Manuscript revised December 22, 2023.
Manuscript publicized February 9, 2024.

[†]The authors are with Graduate School of Science and Technology, Keio University, Yokohama-shi, 223–8522 Japan.
a) E-mail: hoshino@arc.ics.keio.ac.jp
b) E-mail: kawakami@arc.ics.keio.ac.jp
c) E-mail: matutani@arc.ics.keio.ac.jp

DOI: 10.1587/transinf.2023EDP7176

*An early stage of this work appeared in our workshop paper [6]. In this paper, we experimentally demonstrate that the proposed approach can aggregate models with different iteration counts. It is compared to a federated learning approach that uses knowledge distillation.

ations with different hyper-parameters are also conducted. Section 6 concludes this paper.

2. Related Work

2.1 Federated Learning

Federated Averaging (FedAvg) is a basic federated learning algorithm proposed in [1]. Algorithm 1 shows the server- and client-side flows, where K is the total number of clients, k is their index, and $\mathcal{P}_k$ is data at client k. Also, B is the size of a local mini-batch, E is the number of epochs to be trained by each client, and η is a given learning rate. In this algorithm, the first step is to initialize global weight parameters of the model. Then, m clients are randomly selected from K clients, and the server sends the global parameters to the selected clients. The size of m is determined by the client participating rate r. The weight parameters are updated at each epoch (E epochs in total) by each client based on the formula in line 13. After E updates, the clients send their trained local parameters to the server. The server aggregates the received local parameters by taking the average based on the formula in line 8, where n is the total number of data and n_k is the total number of data at client k. The aggregated parameters are then sent back to the clients as global parameters. The above steps are repeated t rounds.

Many federated learning technologies have been studied since FedAvg was proposed in 2016. These technologies are surveyed in [7]. Data heterogeneity is one of important research challenges in these studies since it is a major cause of accuracy degradation. For instance, since a local model is optimized toward the local optima by the client, it may be distant from other clients. Thus, their averaged global model may be far from a part of clients. To deal with this problem, FedProx [8] uses an additional proximal term to limit the number of local updates, and SCAFFOLD [9] uses a variance reduction to correct local updates. These algorithms aim to improve the local training step of FedAvg. In contrast, Personalized Federated Averaging [10] and Adaptive Personalized Federated Learning [11] aim to make personalized models that can achieve good accuracy in local clients.

Algorithm 1 Federated Averaging [1]

1: **function** ExecuteServer()
2: Initialize w_0
3: **for each** round $t = 1, 2, \ldots$ **do**
4: $m \leftarrow \max(r \cdot K, 1)$
5: $S_t \leftarrow$ (random set of m clients)
6: **for each** client $k \in S_t$ in parallel **do**
7: $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
8: $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$
9: **function** ClientUpdate(k, w) ▷ *Run on client k*
10: $\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
11: **for each** local epoch i from 1 to E **do**
12: **for each** batch $b \in \mathcal{B}$ **do**
13: $w \leftarrow w - \eta \nabla \ell(w; b)$

Another challenge is the clients' model heterogeneity. Since not all clients always have the same compute resources, selecting a proper model for each client can help the client heterogeneity. FedFeNN [12] and FedDF [13] address the model heterogeneity. In FedHeNN, each client trains its own model but pulls the representations learned by different clients closer by adding a proximal term to the client's loss function [12]. FedDF is a federated learning algorithm that utilizes a knowledge distillation at the model aggregation step of the federated learning server. In the knowledge distillation step at the server, instead of averaging local parameters received from clients, a batch of sample data is predicted by the local parameters and their output logits are averaged. The averaged logits are then used for updating the client models at the server. Although FedDF allows a federated leaning of different model architectures, model training is needed at the aggregation step of the server in addition to local training at the clients. This means that server-side samples are needed for the server-side knowledge distillation. In [14], a model agnostic federated learning approach that aims to improve participant model performance through learning from other participants via public dataset is proposed. It also relies on public dataset.

2.2 ResNet and Neural ODE

ResNet [3] is a well-known neural network architecture that can increase the number of stacked layers or building blocks by introducing shortcut connections. Using a shortcut connection, an input feature map to a building block is temporarily saved, and then it is added to the original output of the building block to generate the final output of the block. In this paper, one building block in ResNet is called ResBlock.

ODE is composed of an unknown function and its ordinary derivatives. To obtain an approximate numerical solution, an ODE solver such as the first-order Euler method and higher-order Runge-Kutta method can be used. Based on a similarity between the network structure with shortcut connections and the ODE solver, one building block can be interpreted as one step in the ODE solver as suggested in [2]. Assuming that the Euler method is used as an ODE solver, it can be interpreted that the first-order approximation is applied to solve an output of the building block. In this paper, one building block is called ODEBlock, and the whole network architecture consisting of ODEBlocks is called ODENet.

2.3 Depthwise Separable Convolution

CNN is composed of multiple layers, such as convolutional layers, pooling layers, and fully-connected layers. Although CNN achieves a good accuracy in image recognition tasks, each convolutional layer has many parameters. Let N, M, and N_K be the number of input channels, the number of output channels, and the kernel size of one side, respectively. The number of parameters in one convolutional layer is NMN_K^2.

Depthwise separable convolution [5] divides this convolutional layer into two convolutional steps: depthwise convolutional step and pointwise convolutional step. In the depthwise convolutional step, a convolutional operation involving only spatial direction (the size is N_K^2) is applied for each input feature map. Different weight parameters are used for each of N input channels; thus its weight parameter size is NN_K^2. Then, an output feature map of the depthwise convolutional step is fed to the pointwise convolutional step as an input. A 1×1 convolutional operation is applied for each input feature map and for each output channel; thus its weight parameter size is NM. The weight parameter size of the depthwise separable convolution is $NN_K^2 + NM$ in total, which is approximately N_K^2 times reduction, assuming that $N, M \gg N_K$.

As a low-cost CNN model, dsODENet [4] applies the depthwise separable convolution to convolutional layers of ODEBlocks in order to further reduce the parameter size. It was originally proposed to be implemented on resource-limited FPGA (Field-Programmable Gate Array) devices [4]. In this paper, we use dsODENet as a federated learning model architecture in addition to ODENet. Their detailed structures are illustrated in the next section.

3. Proposed Federated Learning

In the traditional federated learning such as FedAvg, the server and all clients have to use the same model. For example, models with different layer depths cannot be averaged when ResNet is used as a model architecture. However, in a real environment, client devices are not the same and are likely to have different compute resources, such as memory capacity and computation power. Since the traditional federated learning cannot aggregate models with different depths, a common model used by all the clients should be carefully selected. In addition, it is necessary to reduce the communication size involved in exchanging weight parameters between the server and clients. In this paper, we use ODENet and dsODENet to enable a flexible federated learning between models with different layer iteration counts and significantly reduce the communication size. These target models are illustrated in Sect. 3.1. We discuss the feasibility of the proposed federated learning in Sects. 3.2 and 3.3.

3.1 Target Models

In this section, we first illustrate the structures and sizes of ResNet and ODENet. Then, we illustrate dsODENet.

Figures 1 and 2 show basic structures of ResNet and the corresponding ODENet. They consist of seven blocks including conv1 and fc. In the ResNet model, conv1 performs convolutional operations as a pre-processing layer, and fc is a post-processing fully-connected layer. After the conv1, C physically-stacked ResBlocks are executed in block1. block2_1 is a downsampling ResBlock to reduce the feature map size, and C physically-stacked ResBlocks on the output of block2_1 are executed in block2_2. The same operation is performed for block3_1 and block3_2 too.

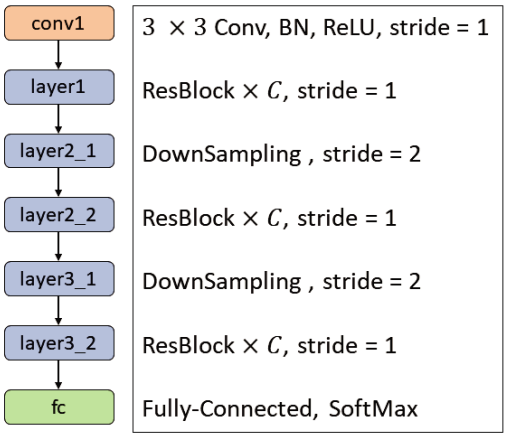


Fig. 1 Structure of 7-block ResNet

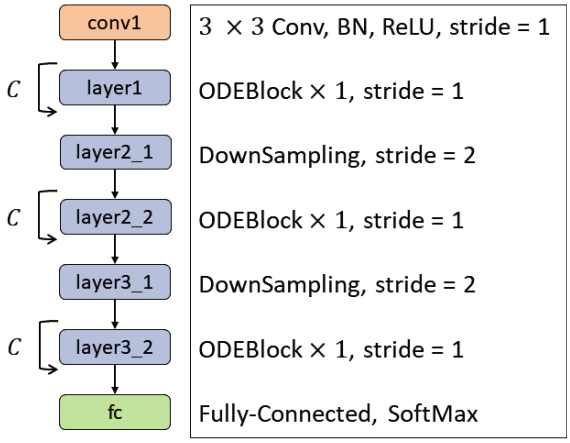
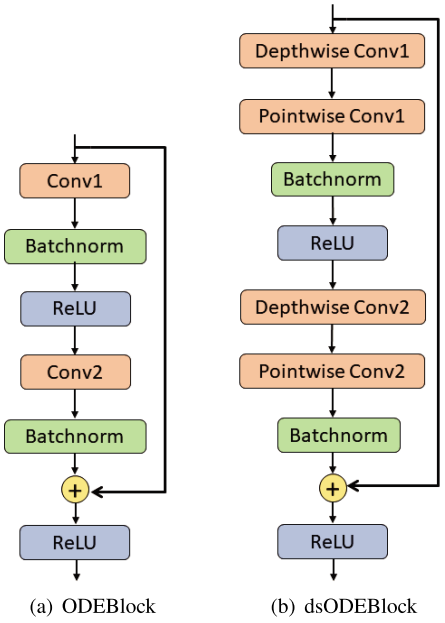


Fig. 2 Structure of 7-block ODENet

ODENet replaces ResBlocks in Fig. 1 with ODEBlocks as shown in Fig. 2. In ResNet, C ResBlocks are physically-stacked in block1, block2_2, and block3_2, while ODENet replaces these C ResBlocks with a single ODEBlock. Instead, ODEBlock is executed C times in block1, block2_2, and block3_2. A downsampling ODEBlock is executed only once in block2_1 and block3_1, respectively.

Here, we analyze the numbers of parameters of ResNet and ODENet. Let $O(L)$ be the number of parameters in one ResBlock and ODEBlock. In ResNet, C ResBlocks are stacked in one block, so the number of parameters is $O(CL)$. In contrast, ODENet repeats ODEBlock C times in one block, so the number of parameters is $O(L)$. The parameter size reduction by ODENet becomes large as C is increased. As shown, the communication size can be reduced by using ODENet as a federated learning model instead of ResNet.

In addition, the number of parameters can be further reduced by using dsODENet [4] as mentioned in Sect. 2.3. Figure 3 (a) shows a structure of an ODEBlock in ODENet. Figure 3 (b) shows that of dsODEBlock in dsODENet, in which each convolutional layer of the ODEBlock is replaced



(a) ODEBlock (b) dsODEBlock

Fig. 3 Structures of ODEBlock and dsODEBLock in ODENet and dsODENet

Table 1 Accuracy of ODENet models trained and tested with same or different depths

Trained as	Teseted as	Accuracy [%]
ODENet-34	ODENet-34	75.46 ± 0.41
	ODENet-50	75.36 ± 0.43
	ODENet-101	75.26 ± 0.46
ODENet-50	ODENet-34	76.13 ± 0.42
	ODENet-50	76.02 ± 0.42
	ODENet-101	75.91 ± 0.52
ODENet-101	ODENet-34	76.25 ± 0.45
	ODENet-50	76.14 ± 0.47
	ODENet-101	76.13 ± 0.45

Table 2 Accuracy of dsODENet models trained and tested with same or different depths

Trained as	Tested as	Accuracy [%]
dsODENet-34	dsODENet-34	71.67 ± 0.53
	dsODENet-50	71.74 ± 0.68
	dsODENet-101	71.45 ± 0.68
dsODENet-50	dsODENet-34	72.24 ± 0.61
	dsODENet-50	72.10 ± 0.62
	dsODENet-101	72.00 ± 0.69
dsODENet-101	dsODENet-34	72.28 ± 0.64
	dsODENet-50	72.11 ± 0.62
	dsODENet-101	72.13 ± 0.64

with two convolutional steps: the depthwise convolutional step and the pointwise convolutional step. Conv1 and Conv2 are convolutional layers, and ReLU (Rectified Linear Unit) is an activation function. This modification can reduce the model and communication sizes compared with ODENet.

3.2 Weight Compatibility with Different Depths

In the case of ResNet, models with different depths have different numbers of stacked ResBlocks (i.e., different C) in their block1, block2_2, and block3_2. If we use FedAvg for federated learning, basically these different ResNet models cannot be averaged at the server due to the model incompatibility. In the case of ODENet, one ODEBlock is repeated C times in block1, block2_2, and block3_2 of ODENet. In other words, ODENet models with different depths differ only in the numbers of iterations of ODEBlocks, not in the number of ODEBlocks. Therefore, the structure of the ODENet models with different C is the same, so their parameters can be averaged at the server. Using ODENet as a federated learning model enables a flexible federated learning between models with different iteration counts and fully utilizes the performance of each client device. dsODENet [4] also enables such a flexible federated learning between models with different iteration counts for the same reason as ODENet.

Here, we examine if the above observation can work. This section focuses on the weight parameter compatibility of models which have different C to demonstrate the feasibility of the proposed federated learning. Specifically, inference accuracies of ODENet and dsODENet models which were trained for the same or different C are evaluated. Please note that, in the following, we use the total number of executed convolutional layers (denoted as N) to represent the depths of the ResNet, ODENet, and dsODENet models. We assume $N = 6C + 6$ in this experiment[†] and use $N = 34$, 50, and 101. For example, the inference accuracy of ODENet-34 is evaluated using weight parameters trained as ODENet-50. We selected two models from $N = 34$, 50, and 101: one for training and another for inference. The experiment is performed 100 times for each combination. CIFAR-10 dataset [15] is used for the training and inference. Here, we consider all the combinations of two models from ODENet-34, ODENet-50, and ODENet-101 (e.g., ODENet-34 and ODENet-50). One of the two models (e.g., ODENet-34) is trained with the whole CIFAR-10 dataset, and then the trained weight parameters are used and tested as another model (e.g., ODENet-50). The same experiment is also performed for dsODENet.

Tables 1 and 2 show the inference accuracy of every combination. Figure 4 shows box-plots of accuracies of ODENet-34, 50, and 101 using weight parameters trained as ODENet-50. Figure 5 shows those of dsODENet. The results from Fig. 4 and Table 1 show that the accuracies are almost the same regardless of the tested models if the trained model is the same. The results from Fig. 5 and Table 2 also show the same tendency in dsODENet. Figure 6 shows box-plots of accuracies of ODENet-50 using weight parameters trained as ODENet-34, 50, and 101, respectively. Figure 7 shows those of dsODENet-50. The results from Fig. 6 and Table 1 show that the accuracies depend on N of the trained model. The results from Fig. 7 and Table 2 show the same tendency in dsODENet.

[†]In Fig. 1, ResBlocks are executed $3C + 2$ times, each contains two convolutional layers. The pre- and post-processing (conv1 and fc) layers are also included in N; thus $N = 2(3C + 2) + 2$.

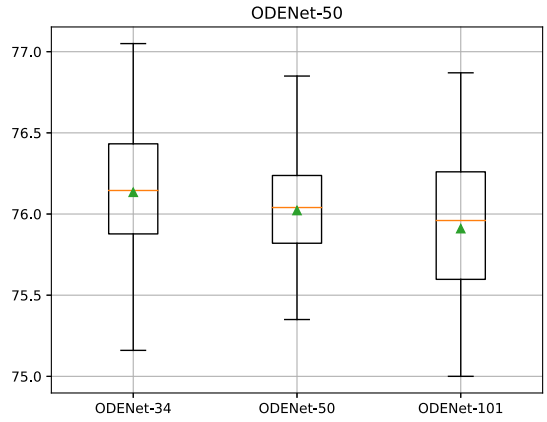


Fig. 4 Accuracy of ODENet-34, 50, and 101 using parameters trained as ODENet-50 [%]

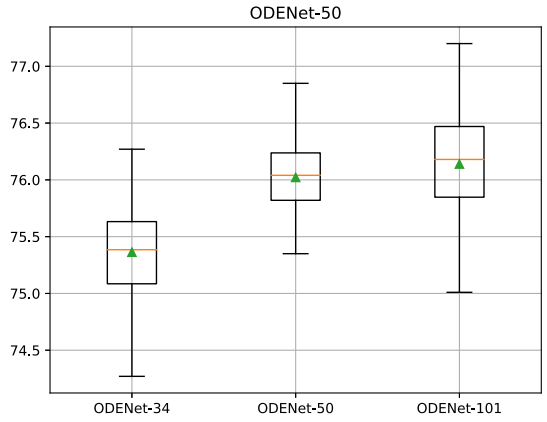


Fig. 6 Accuracy of ODENet-50 using parameters trained as ODENet-34, 50, and 101 [%]

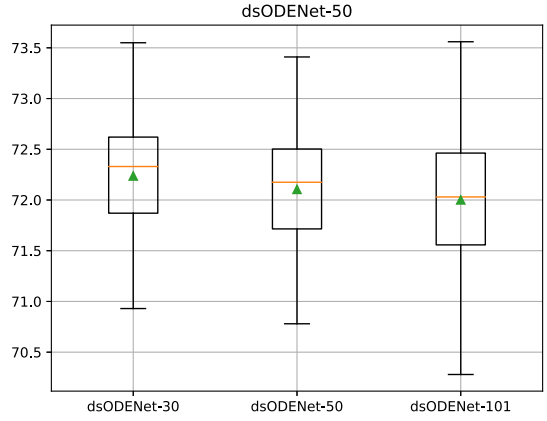


Fig. 5 Accuracy of dsODENet-34, 50, and 101 using parameters trained as dsODENet-50 [%]

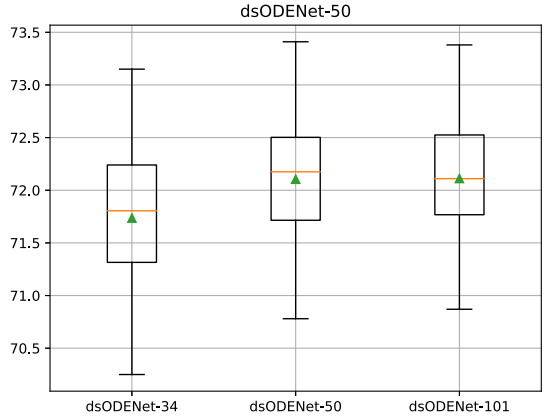


Fig. 7 Accuracy of dsODENet-50 using parameters trained as dsODENet-34, 50, and 101 [%]

As mentioned above, the inference accuracies of the ODENet and dsODENet models are reasonable even if the trained N and tested N are different. This means that ODENet and dsODENet models have a weight parameter compatibility with different depths. Furthermore, Table 1 shows that the accuracies of models that were trained as ODENet-101 are higher than those that were trained as ODENet-34 and ODENet-50. Table 2 also shows a similar tendency in dsODENet. These results indicate that using a deeper model for training can help to enhance the accuracy in ODENet and dsODENet models.

3.3 Federated Learning with Different Depths

In Sect. 3.2, we showed the weight parameter compatibility in ODENet and dsODENet models with different depths. In this section, we examine the feasibility of federated learning between ODENet models with different depths. Specifically, we perform federated learning of two ODENet models from among N = 34, 50, and 101 to see if it works correctly. In this experiment, FedAvg is used as a federated learning algorithm. The number of clients K is only 2, the client participating rate r is 1, the number of local epochs E is 5, and the number of communication rounds is 20. In this experiment, the whole CIFAR-10 dataset is randomly partitioned into two clients evenly (i.e., not biased). ODENet-50 is used as a global model regardless of the combinations of two models. The same experiment is performed for dsODENet too.

Figures 8 and 9 show training curves (epoch number vs. loss value) of the ODENet and dsODENet models, respectively. A loss value in Fig. 8 shows a test loss computed by a global model (i.e., ODENet-50) which has been aggregated from two clients. In Fig. 8, the black line shows the loss values when federated learning is performed between ODENet-50 and ODENet-50. The red line shows those between ODENet-34 and ODENet-50. The green line shows those between ODENet-50 and ODENet-101. The blue line shows those between ODENet-34 and ODENet-101. Figure 9 shows experimental results for dsODENets. Meanings of line colors are the same as those in Fig. 8 but models used are dsODENets. The horizontal axis of these figures represents the number of epochs, and the vertical axis represents the loss value. As shown, the loss values decrease as the number of epochs is increased, and then they are converged around 50 epochs in all the combinations. This indicates that

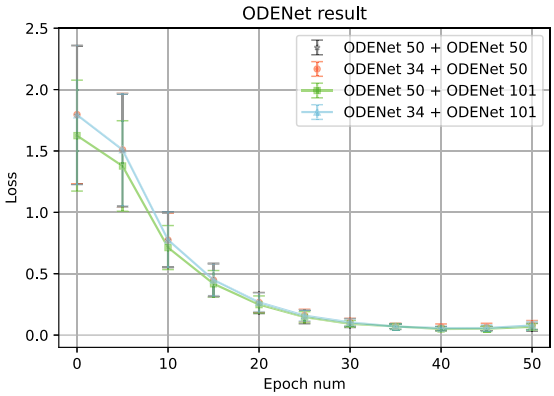


Fig. 8 Training curve (epoch number vs. loss value) of ODENet

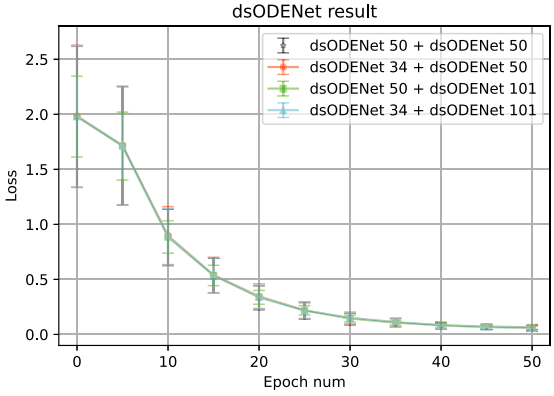


Fig. 9 Training curve (epoch number vs. loss value) of dsODENet

these model combinations of different depths can be trained successfully. This and previous sections demonstrated that ODENet and dsODENet are capable of federated learning between models with different depths.

4. Evaluations

In this section, we evaluate the proposed flexible federated learning approach that uses FedAvg as a federated learning algorithm and ODENet, dsODENet, and ResNet as client models. They are compared with FedDF which is a model agnostic federated learning approach using a knowledge distillation. Finally, the proposed approach is evaluated in terms of the model size and communication cost.

Python 3.8.5, PyTorch 1.8.1 [16], and torchvision 0.9.1 are used for the model implementation. A machine with Ubuntu 18.04.5 LTS (64-bit), Intel Core i7-10700K CPU @ 3.8GHz, 32GB DDR4 SDRAM, and NVIDIA GeForce RTX 3090 GPU is used for the evaluation in this paper.

4.1 Accuracy

Although Sect. 3.3 showed the feasibility of the proposed federated learning, only two clients were used in the experiments. However, it is expected that more clients participate in a practical federated learning scenario. In this section, we increase the number of clients and conduct additional experiments.

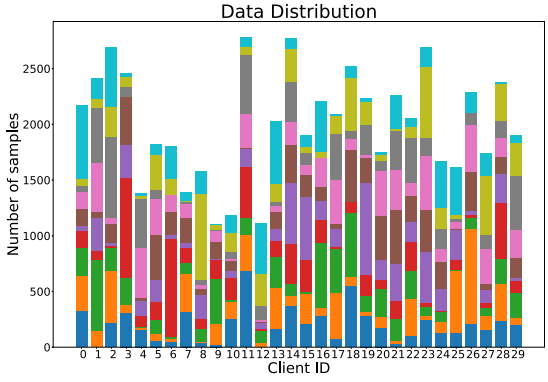


Fig. 10 Data distribution using Dirichlet distribution (α = 1)

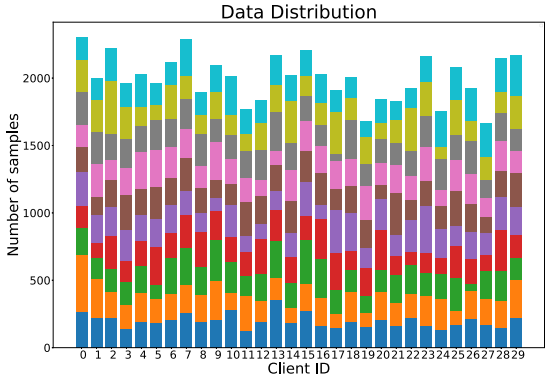


Fig. 11 Data distribution using Dirichlet distribution (α = 10)

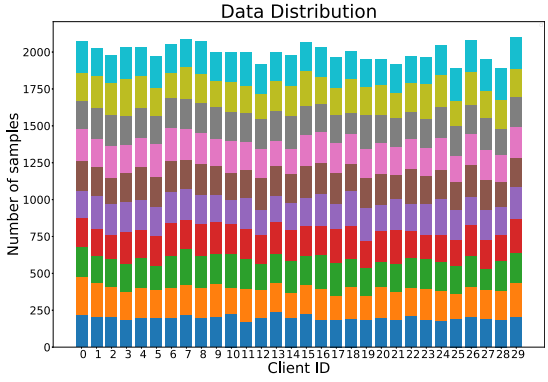


Fig. 12 Data distribution using Dirichlet distribution (α = 100)

In addition, it is expected that there is a bias in the data distribution for each client in the case of real environments. This means that the data distribution for each client is non-iid. In this experiment, Dirichlet distribution is thus used to make non-iid data environments. Dirichlet distribution is a kind of continuous multivariate probability distributions, and its data distribution is controlled by a vector α. We use α = 1, 10, and 100. CIFAR-10 is used as a dataset. Figures 10–12 show examples of data distributions with α = 1, 10, and 100 as Dirichlet distribution parameters, respectively.

Table 3 Accuracy of ODENet and dsODENet when using FedAvg and FedDF (α = 1)

Model	Algorithm	Top5 [%]	Top1 [%]
ODENet	FedAvg	97.02	69.03
	FedDF	98.23	73.73
dsODENet	FedAvg	97.24	67.66
	FedDF	98.40	75.67

Table 4 Accuracy of ODENet and dsODENet when using FedAvg and FedDF (α = 10)

Model	Algorithm	Top5 [%]	Top1 [%]
ODENet	FedAvg	97.02	70.47
	FedDF	98.87	78.55
dsODENet	FedAvg	97.34	69.82
	FedDF	98.81	78.63

Table 5 Accuracy of ODENet and dsODENet when using FedAvg and FedDF (α = 100)

Model	Algorithm	Top5 [%]	Top1 [%]
ODENet	FedAvg	97.31	70.72
	FedDF	98.78	78.67
dsODENet	FedAvg	97.24	70.64
	FedDF	98.89	78.72

In this experiment, the number of clients K is 30, the number of local epochs E is 40, and the number of communication rounds is 100. Among the 30 clients, the numbers of clients that use ODENet-34, ODENet-50, and ODENet-101 are 10, 10, and 10, respectively. ODENet-50 is used as a global model. In Algorithm 1, r is the client participating rate in the aggregation at each communication round. In this experiment, r is set to 0.2, which means that six models are randomly selected from the 30 clients. FedAvg is used in the proposed federated learning approach, and the results are compared with those of FedDF.

Tables 3–5 show the accuracies of ODENet-50 and dsODENet-50 with FedAvg and FedDF when α = 1, 10, and 100, respectively. Top5 accuracies are mostly high regardless of α in both the models. Top1 accuracies are decreased when α is small (e.g., α = 1). In this case, the data distribution is highly biased, and the biased data distribution negatively affects the accuracy. Although addressing the data heterogeneity is a crucial challenge in federated learning, many studies have been conducted to overcome this issue as mentioned in [7] and thus addressing this is beyond the scope of this paper.

When we compare the proposed approach using FedAvg with FedDF, the accuracies of the proposed approach are lower than those of FedDF. This result is reasonable since FedDF introduces additional training overheads for the knowledge distillation (e.g., prediction and training of client models) at the server in addition to local training at the clients while the proposed approach does not impose such overheads as well as the conventional FedAvg based approach. To bring the evaluation condition of FedDF closer to that of the proposed approach, here we limit the number of samples to be used in the knowledge distillation at the server.

Table 6 shows the evaluation results of FedDF when the number of server-side samples available for the knowledge distillation is limited. ∞ means that there is no limitation in the number of server-side samples; that is, the whole CIFAR-10 training dataset can be used for the knowledge distillation in FedDF. However, please note that using the whole CIFAR-10 training dataset for the knowledge distillation significantly increases the training time of FedDF. We thus reasonably stop the knowledge distillation if test accuracy is not improved during the latest 10 training batches assuming that the batch size is 100. α is set to 10 as the data distribution parameter. The other conditions are same as those in Table 4. As shown in Table 6, the accuracy of FedDF increases as the number of server-side samples used in the knowledge distillation is increased. When the number of the available server-side samples is small, the accuracy of FedDF becomes close to (or lower than) the proposed approach as shown in Tables 4 and 6. Please note that "Time" in Table 6 represents the average time interval between two successive communication rounds. Such server-side overheads of FedDF are discussed in Sects. 5.1 and 5.2.

Table 6 Accuracy of FedDF when varying the number of samples used for knowledge distillation (α = 10)

Model	# of server samples	Top5 [%]	Top1 [%]	Time [s]
ODENet	5	97.22	69.53	427.89
	10	97.09	70.49	439.03
	100	97.95	74.37	460.23
	500	98.57	76.48	693.21
	1000	98.71	78.32	760.04
	∞	98.87	78.55	1406.73
dsODENet	5	97.18	69.52	427.98
	10	97.16	69.82	434.77
	100	97.93	73.88	466.62
	500	98.41	76.89	699.93
	1000	98.73	78.32	767.08
	∞	98.81	78.63	1454.90

4.2 Memory and Communication Sizes

Table 7 compares ResNet, ODENet, and dsODENet in terms of the model size, the forward/backward pass size, and the total memory size for training. "Model size" means the necessary memory size to retain the model. "Memory size during training" includes the model size, the forward/backward pass size, and the input/output data sizes. These values are measured by using torchinfo which is a tool that reports the parameter size information.

The communication is required between the server and clients in each communication round. In Table 7, "# of parameters transferred" is the sum of the weight parameters of convolutional layers and fully-connected layers in these models. In the following discussion, it is simply denoted as communication size. Table 7 shows that both the ODENet and dsODENet models reduce the communication sizes compared with the corresponding ResNet model. Compared with ResNet, the communication size of ODENet-50 is 10.6% of the original ResNet model. In the case of ResNet and dsODENet, the communication size of dsODENet-50 is 5.3% of the ResNet model. These results show that the

Table 7 Memory and communication sizes of ResNet, ODENet, and dsODENet

Model	# of parameters transferred	Model size [MB]	Forward/backward pass size [MB]	Memory size during training [MB]
ResNet-34	21,797,672	87.19	61.43	149.24
ResNet-50	25,557,032	102.23	181.85	284.70
ResNet-101	44,549,160	178.20	265.41	444.22
ODENet-34	2,719,434	10.88	77.83	89.32
ODENet-50	2,719,434	10.88	135.17	146.66
ODENet-101	2,719,434	10.88	203.98	215.48
dsODENet-34	1,344,128	5.38	106.84	112.83
dsODENet-50	1,344,128	5.38	193.19	199.18
dsODENet-101	1,344,128	5.38	296.81	302.80

use of ODENet and dsODENet can significantly reduce the communication size between the server and clients. The communication size increases as N is increased in ResNet. On the other hand, communication sizes are constant regardless of N in the cases of ODENet and dsODENet. This is because the number of physically-stacked blocks is the same in ODENet and dsODENet even if N is different, and only the number of iterations of each block is different.

Please note that, since the model size is small in ODENet and dsODENet, the required memory capacity can also be reduced by these models compared with the original ResNet model. As shown in Table 7, the memory sizes to train ODENet-50 and dsODENet-50 models are 52% and 70% of that of ResNet-50, respectively. This result demonstrates that our proposed approach is beneficial in terms of the memory size for the training.

5. Discussions

Here, we discuss pros and cons of the proposed approach against counterparts. Additional evaluations with different hyper-parameters are also conducted.

5.1 Overall Comparisons

Here, we compare the proposed approach with ResNet + FedAvg and ResNet + FedDF in terms of the accuracy, communication size, and computation time. As for the comparison to ResNet + FedAvg, since FedAvg cannot aggregate ResNet models with different depths, we compare the proposed approach with the ResNet + FedAvg cases where all the clients have the same ResNet model. We assume that 30 clients join the federated learning where 10, 10, and 10 clients have ResNet-34, ResNet-50, and ResNet-101 models, respectively. The evaluation conditions are the same as those of Tables 3, 4, and 5. ResNet-50 is used as a global model. The number of communication rounds is 100, and the number of local epochs E is 40. α is set to 10. The client participating rate r is 0.2.

Table 8 shows the comparison results. In this table, "Time" represents the average time interval between two successive communication rounds. "Communication size" represents the average parameter size that a single client sends to the server in a single communication round. "Different depth" indicates whether models with different depths can be aggregated. Although baseline accuracies of ODENet and dsODENet are inherently lower than ResNet regardless of the federated learning algorithms (FedAvg and FedDF), the time interval of our proposed approach is mostly shorter than the other solutions. Also, our approach is advantageous in terms of the communication size. Another benefit of our approach against FedDF is that it does not require server-side samples, while it can aggregate models with different depths, which is a benefit against ResNet + FedAvg.

5.2 Comparisons to FedDF

As shown in Tables 3, 4, and 5, FedDF presents a higher accuracy compared to FedAvg. However, as shown in Table 6, accuracy of FedDF becomes comparable to our proposed approach if 5 or 10 server-side samples are only available. In addition, in FedDF, clients need to upload their models to the server [13], and thus its communication size is comparable to that of FedAvg. Please note that FedDF can improve the accuracy by increasing the number of server-side samples for the server-side knowledge distillation. As shown in Table 6, the time interval between two successive communication rounds of FedDF increases as the number of server-side samples increases, and this time interval becomes longer than that of our proposed approach. One of important benefits of the classic federated learning is that the server does not have to retain privacy-sensitive training samples. The simplicity of the server-side processing would also be a benefit. On the other hand, FedDF requires the server-side training samples. We need to consider cases where concrete or accurate server-side training samples which can be uploaded to cloud servers may not be available. Our approach does not require server-side samples, while it can aggregate models with different depths, which is a benefit against ResNet + FedAvg. As shown in Table 7, the model and training memory sizes of ODENet and dsODENet are smaller than those of ResNet. Actually, a smaller memory size can help the client heterogeneity since clients with a tighter resource limitation can join the federated learning. As such, in our approach, clients with rich compute resources can use deeper models, while those with limited resources can use shallower models.

5.3 Unbalanced Client Heterogeneity

In Sect. 4.1, 30 clients use one of three models with different depths (e.g., ODENet-34, ODENet-50, and ODENet-101). Ratios of these three models are balanced; that is, 10 clients,

Table 8 Overall comparisons of proposed approach and counterparts (α = 10)

	Different depths	# of server samples	Top5 accuracy [%]	Top1 accuracy [%]	Time [s]	Communicatino size [MB]
ResNet-34 + FedAvg		0	99.17	85.36	382.65	87.19
ResNet-50 + FedAvg		0	99.50	87.70	844.97	102.23
ResNet-101 + FedAvg		0	99.37	87.37	1983.56	178.20
ODENet + FedAvg	$\surd$	0	97.02	70.47	420.69	10.88
dsODEnet + FedAvg	$\surd$	0	97.34	69.82	420.11	5.38
ResNet + FedDF	$\surd$	10	98.76	80.63	628.40	122.54
ODENet + FedDF	$\surd$	10	97.09	70.49	439.03	10.88
dsODENet + FedDF	$\surd$	10	97.16	69.82	434.77	5.38

Table 9 Evaluation results of proposed approach in weak-biased case

Model	α	Top5 [%]	Top1 [%]
ODENet	1	97.81	70.39
	10	97.86	73.48
	100	97.67	73.67
dsODENet	1	97.14	68.96
	10	97.71	73.14
	100	97.84	73.09

Table 10 Evaluation results of proposed approach in strong-biased case

Model	α	Top5 [%]	Top1 [%]
ODENet	1	97.34	70.90
	10	97.73	73.48
	100	97.53	73.39
dsODENet	1	97.35	71.92
	10	97.53	73.34
	100	97.66	73.05

10 clients, and 10 clients use ODENet-34, ODENet-50, and ODENet-101, respectively. Here, we conduct experiments of unbalanced cases. Specifically, we examine two cases: 1) 24 clients use ODENet-34, three clients use ODENet-50, and three clients use ODENet-101; 2) three clients use ODENet-34, three clients use ODENet-50, and 24 clients use ODENet-101. The former case is referred to as "weak-biased" and the latter case is referred to as "strong-biased". The other evaluation conditions are the same as those in Sect. 4.1. Tables 9 and 10 show the evaluation results of these two cases. As shown in Tables 9 and 10, accuracy of the strong-biased case is slightly higher than that of the weak-biased case.

5.4 Homogenous Model Case

Here, we conduct experiments where all the clients have the same model, which is an ideal situation. Table 11 shows the results. α is set to 10. The other evaluation conditions are the same as those in Sect. 4.1. As shown in Table 11, although ODENet and dsODENet can significantly reduce the number of parameters (please see "Model size") compared to ResNet, ODENet and dsODENet show a lower accuracy compared to ResNet. This result is reasonable since the numbers of parameters of ODENet and dsODENet are significantly small compared to ResNet. Also, the time intervals of ODENet and dsODENet are shorter than those of ResNet. When all the clients have the same model, the time intervals of ODENet and dsODENet are shorter than those of heterogenous client cases shown in Table 8. However, our proposed approach can aggregate models with different depths, while ResNet cannot allow such a heterogeneity, which is a benefit of our proposed approach.

Table 11 Evaluation results when all clients have same model (α = 10)

Model	Top-5 [%]	Top-1 [%]	Time [s]	Model size [MB]
ODENet-34	98.06	74.51	380.70	10.88
ODENet-50	98.00	74.53	381.94	10.88
ODENet-101	98.09	74.55	384.83	10.88
dsODENet-34	98.02	74.74	377.95	5.38
dsODENet-50	98.03	74.49	381.19	5.38
dsODENet-101	98.03	74.05	383.28	5.38
ResNet-34	99.17	85.36	382.65	87.19
ResNet-50	99.50	87.70	844.97	102.23
ResNet-101	99.37	87.37	1983.56	178.20

Table 12 Evaluation results when the number of local epochs E is different for each client

Model	α	Top5 [%]	Top1 [%]	Time [s]	Model size [MB]
ODENet-34	1	97.93	73.77	432.76	10.88
	10	98.13	75.94	433.69	10.88
	100	98.21	76.10	435.02	10.88
ODENet-50	1	97.88	73.92	438.62	10.88
	10	98.21	75.62	433.94	10.88
	100	98.23	75.68	437.02	10.88
ODENet-101	1	98.02	73.92	435.57	10.88
	10	97.98	75.55	436.53	10.88
	100	98.21	76.24	437.61	10.88

5.5 Heterogeneous Epoch Number

As an alternative to our approach, customizing the number of local epochs E for each client depending on the compute resource of each client would help the client heterogeneity. For example, we can increase the number of epochs for stronger clients, while we can decrease the number of epochs for weaker clients. Here, we conduct this experiment. Table 12 shows the results. We use CIFAR-10 dataset. FedAvg is used for the federated learning algorithm. The number of communication rounds is 100. The number of clients K is 30. r is set to 0.2; thus, six clients join the federated learning. The important difference to our proposed approach is that all the clients have the same model, while the number of epochs is different for each client. Specifically, the number of local epochs E is randomly selected from 20, 40, and 80 for each client. The selected epoch number is statically fixed; thus, it is not changed during the federated learning.

As shown in Table 12, the accuracy of this heterogeneous epoch number approach is higher than our proposed approach. However, regarding the client heterogeneity, the heterogeneous epoch number approach can address the CPU

Table 13 Evaluation results when the number of clients K is 50

Model	Algorithm	α	Top-1 [%]
ODENet	FedAvg	1	67.21
		10	69.98
		100	70.38
dsODENet	FedAvg	1	67.42
		10	70.28
		100	70.87

Table 14 Evaluation results when participating client rate r is 0.1

Model	Algorithm	α	Top-1 [%]
ODENet	FedAvg	1	65.94
		10	69.47
		100	70.17
dsODENet	FedAvg	1	65.53
		10	70.80
		100	71.24

performance heterogeneity of clients since the number of epochs can be tuned depending on the CPU performance of each client. On the other hand, our proposed approach can customize the model depths depending on the CPU performance and memory size of clients; thus, our approach can address the client heterogeneity of both the CPU performance and memory capacity of clients. We believe that this heterogeneous epoch number approach can be combined with our proposed approach, and exploring this possibility is our future work.

5.6 Different Client Number and Participating Rate

We conduct an additional experiment in which the number of clients K is increased to 50, in which 10 clients use ODENet-34, 30 clients use ODENet-50, and 10 clients use ODENet-101. Table 13 shows the results. In addition, we conduct another experiment in which the client participating rate r is 0.1. Table 14 shows the results. In both the cases, the results are consistent with the results presented in Sect. 4.1.

6. Conclusions

In this paper, we proposed a flexible federated learning approach that can aggregate models with different iteration counts by utilizing ODENet and dsODENet as federated learning models. We demonstrated that these models with different iteration counts can be aggregated correctly (i.e., having the weight compatibility) in the cases of ODENet and dsODENet. Then, the proposed approach simply using FedAvg was compared with FedDF in terms of the accuracy. The experiment results showed that the higher accuracy of FedDF come from additional knowledge distillation overheads at the server. On the other hand, the proposed approach can simply aggregate models with different iteration counts without the server-side training nor uploading training samples to the server. In addition, ODENet and dsODENet were evaluated in terms of the model and communication sizes. Compared with ResNet-50, ODENet-50 and dsODENet-50 successfully reduced the communication sizes by 89.4% and by 94.7%, respectively. These results showed that our approach can significantly reduce the communication overhead while enabling the aggregation of models with different iteration counts.

As a future work, we will evaluate the feasibility of federated learning with ANODE [17] which is a Neural ODE based approach that utilizes a checkpointing method. We are also planning to improve accuracy when α is small. We will combine our approach with state-of-the-art federated learning algorithms.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B.A.Y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," Proc. International Conference on Artificial Intelligence and Statistics (AISTATS), pp.1273–1282, April 2017.
[2] R.T.Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural Ordinary Differential Equations," Proc. Annual Conference on Neural Information Processing Systems (NeurIPS), pp.6572–6583, Dec. 2018.
[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.770–778, June 2016.
[4] H. Kawakami, H. Watanabe, K. Sugiura, and H. Matsutani, "A Low-Cost Neural ODE with Depthwise Separable Convolution for Edge Domain Adaptation on FPGAs," IEICE Trans. Inf. & Syst., vol.E106-D, no.7, pp.1186–1197, July 2023.
[5] A.G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications." arXiv Preprint arXiv:1704.04861, April 2017.
[6] Y. Hoshino, H. Kawakami, and H. Matsutani, "Communication Size Reduction of Federated Learning based on Neural ODE Model," Proc. International Symposium on Computing and Networking (CANDAR) Workshops, pp.55–61, Nov. 2022.
[7] Q. Li, Z. Wen, Z. Wu, S. Hu, N. Wang, Y. Li, X. Liu, and B. He, "A Survey on Federated Learning Systems: Vision, Hype and Reality for Data Privacy and Protection," IEEE Trans. Knowl. Data Eng., vol.35, no.4, pp.3347–3366, April 2021.
[8] T. Li, A.K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated Optimization in Heterogeneous Networks." arXiv Preprint arXiv:1812.06127, April 2020.
[9] S.P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A.T. Suresh, "SCAFFOLD: Stochastic Controlled Averaging for Federated Learning," Proc. International Conference on Machine Learning (ICML), pp.5132–5143, July 2020.
[10] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized Federated Learning with Theoretical Guarantees: A Model-Agnostic Meta-Learning Approach," Proc. the Annual Conference on Neural Information Processing Systems (NeurIPS), pp.3557–3568, Dec. 2020.
[11] Y. Deng, M.M. Kamani, and M. Mahdavi, "Adaptive Personalized Federated Learning," arXiv Preprint arXiv:2003.13461, March 2020.
[12] D. Makhija, X. Han, N. Ho, and J. Ghosh, "Architecture Agnostic Federated Learning for Neural Networks," Proc. International Conference on Machine Learning (ICML), pp.14860–14870, July 2022.
[13] T. Lin, L. Kong, S.U. Stich, and M. Jaggi, "Ensemble Distillation for Robust Model Fusion in Federated Learning," arXiv Preprint arXiv:2006.07242, March 2021.
[14] W. Huang, M. Ye, B. Du, and X. Gao, "Few-Shot Model Agnostic Federated Learning," Proc. International Conference on Multimedia (MM), pp.7309–7316, Oct. 2022.
[15] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," April 2009.
[16] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan,

T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," Proc. Annual Conference on Neural Information Processing Systems (NeurIPS), pp.8024–8035, Dec. 2019.

[17] A. Gholaminejad, K. Keutzer, and G. Biros, "ANODE: Unconditionally Accurate Memory-Efficient Gradients for Neural ODEs," Proc. International Joint Conference on Artificial Intelligence (IJCAI), pp.730–736, Aug. 2019.

Yuto Hoshino received the BE degree from Keio University in 2022. He is currently a master course student in Keio University.

Hiroki Kawakami received the BE and ME degrees from Keio University in 2021 and 2023, respectively.

Hiroki Matsutani received the BA, ME, and PhD degrees from Keio University in 2004, 2006, and 2008, respectively. He is currently a professor in the Department of Information and Computer Science, Keio University. His research interests include the areas of computer architecture and machine learning.