

Modular Federated Learning

Kuo-Yun Liang
Connected Systems
Scania CV AB

Södertälje, Sweden
kuo-yun.liang@scania.com

Abhishek Srinivasan
Connected Systems
Scania CV AB

Södertälje, Sweden
abhishek.srinivasan@scania.com

Juan Carlos Andresen
Connected Systems
Scania CV AB

Södertälje, Sweden
juan-carlos.andresen@scania.com

Abstract—Federated learning is an approach to train machine learning models on the edge of the networks, as close as possible where the data is produced, motivated by the emerging problem of the inability to stream and centrally store the large amount of data produced by edge devices as well as by data privacy concerns. This learning paradigm is in need of robust algorithms to device heterogeneity and data heterogeneity. This paper proposes **ModFL** as a federated learning framework that splits the models into a *configuration module* and an *operation module* enabling federated learning of the individual modules. This modular approach makes it possible to extract knowledge from a group of heterogeneous devices as well as from non-IID data produced from its users. This approach can be viewed as an extension of the federated learning with personalisation layers **FedPer** framework that addresses data heterogeneity. We show that **ModFL** outperforms **FedPer** for non-IID data partitions of CIFAR-10 and STL-10 using CNNs. Our results on time-series data with HAPT, RWTHA, and WISDM datasets using RNNs remain inconclusive, we argue that the chosen datasets do not highlight the advantages of **ModFL**, but in the worst case scenario it performs as well as **FedPer**.

Index Terms—Artificial neural network, federated learning, data heterogeneity, device heterogeneity

I. INTRODUCTION

The collective amount of data created in modern life mobile devices, IoT home devices, wearables and other edge devices is practically immeasurable. Often such large amount of data makes it unfeasible to stream, collect and store the data for further processing. On the other hand, traditional machine learning (ML) approaches need to centrally collect the data for model training. Additionally, privacy concerns about data collection and government regulations, such as the European GDPR, have set focus on federated learning [1] as an alternative way of training ML models that cope with such restrictions. This approach trains the ML models close to the network edges and, therefore, close to the data sources. Only the locally trained models are centrally collected and aggregated to a global model. The *De Facto* standard of federated learning **FedAvg** [1] averages the local weights from the devices in the federation to a global model. This empirical approach has shown to be stable even for non-convex optimisation problems and is used as *the benchmark* to compare to for newly developed federated learning protocols.

Even though increasing research attention is shifting towards this field, it is as yet at an early stage and core challenges are still to be solved. The authors of [2] divide them

into four categories; expensive communication, device heterogeneity, data heterogeneity and privacy concerns. Our focus lies in device and data heterogeneity. Device heterogeneity comes from the fact that different edge devices have different hardware and thus a variability in CPU, memory, sensors or network connectivity, leading to a variability of input features, of local training times or communication dropouts. One factor that often is not mentioned as part of this variability is device generation or device model, *i.e.*, edge devices that perform the same task or measure the same observable by collecting different types of inputs. For example, early versions of human activity recording gadgets might only use the accelerometer for classifying the activities, while later versions might have added a gyroscope sensor to improve accuracy. This additional sensor is a challenge to traditional federated learning approaches, as it needs to train from scratch a new model architecture. In the present work we will focus on the generation difference type of device heterogeneity, although this approach can be also used to ease the problem of different training times due to computational power. Data heterogeneity arises as different users (edge devices) in a federation, do in general generate data in a non-independent and non-identically distributed (non-IID) way. In the case of human activity classification, different persons do spend different amount of time walking, running, jogging, jumping, lying or sitting. Some of them might never run or jump, while others might have a regular training schedule.

Recent work addressing the data heterogeneity has proposed federated learning with personalisation layers (**FedPer**) [3]. In this approach, the authors propose to divide the neural networks (NN) into so called base layers and personalisation layers. This approach combats effectively the ill-effects of data heterogeneity. From **FedPer** we borrow the notion of splitting the NN in two modules; a *configuration module* and an *operation module*. We assume a scenario where different models of edge devices have a different configuration of inputs (it can be number and/or different kind of sensors) to perform the same task (measure the same observable). At the same time, the edge devices are owned by different clients that can be categorised by how they operate the edge devices. Our main assumptions are that the configuration modules of models trained on such devices can be trained in a federated learning way between the devices of the same kind, whereas the operation modules can be trained between the same group

of operational clients. We call this concept modular federated learning `ModFL`.

In this work we show that the `ModFL` framework using convolutional neural networks (CNN) for image classification have a higher accuracy than the `FedPer` approach. We show how the proposed `ModFL` protocol can benefit of federation among like-minded clients to improve accuracy even for non-IID data. Moreover, `ModFL` handles device generation heterogeneity by grouping federation groups of same generation devices. For recurrent neural networks (RNN) and time-series data, the results are not conclusive. We argue that more challenging time-series data sets have the potential to show similar accuracy boosts as for the CNN and image classification.

The paper is structured as following: the next section summarises related work to `ModFL`, following by necessary definitions and describes the `ModFL` protocol proposed in this work, then we describe the experiments and results that demonstrate the effectiveness of this protocol. Finally, we give final conclusions and further work in the last section.

II. RELATED WORK

Modularity in NN was investigated by [4] where the authors showed that layered NN can be decomposed in small sets of independent neural networks. They detected units of similar connection patterns. Recent work from [5] shows that multi-layer perceptrons (MLP) neural networks are significantly more modular than random networks with the same distribution of weights when trained with weight pruning. They show that this is not a feature of the datasets it was trained on, but a general property of MLP. The current work is motivated by these results; one question that proceeded the development of `ModFL` framework is how to use this modularity property to train individual sub-modules or sets of independent NN in a federated learning fashion.

The *De Facto* standard of federated learning is federated averaging (`FedAvg`) first introduced in the seminal work by [1]. Here each device optimises the local version of the global model by stochastic gradient descent with the same learning rate and number of local epochs. The resulting local model updates are centrally averaged to a global model. It is important to level the hyperparameters of `FedAvg` to allow as many local epochs as possible to reduce communication cost, on the other hand these local epochs cannot be too long as the local models can reach local optima deviating too much from the global optimum, making convergence slower or even making the method diverge. A proposed framework to reduce communication and increase the local epochs while ensuring convergence is `FedProx` [6], here a proximal term is added to the cost function to be optimised during the learning period. This term helps the local updates stay closer to the global model enabling larger local epochs and thus minimising communication costs. Furthermore, a variable number of local iterations across different edge devices is allowed, making it possible to adjust this number to the device hardware capabilities.

To address data heterogeneity in a federated learning setup, *i.e.*, differences in the data sources, the `FedPer` framework proposed to split the neural networks into *base layers* and *personalisation layers* [3]. The latter ones even after local training remain in the edge devices and are not shared, whereas the former ones will follow collective training frameworks such as the `FedAvg`. In this way each device will have a unique model composed of the *global* base layers and the *local* personalisation layers. The assumption here is that the personalisation layers learn the specific features of the data coming from the personal users habits, in contrast the base layers learn more general features that are common to all users. By splitting and training the network in such a way, the trained models perform better in non-IID data compared to `FedAvg`. In this work, the proposed framework of `ModFL` extends the `FedPer` to also deal with device heterogeneity of edge devices.

Further approaches to personalised federated learning that tackle the heterogenous data problem are `Per-FedAvg` that uses meta-learning to train a global model that has “good” initial values such that users can subsequently adapt the model, within limited computational expences, to better generalise to the user’s data [7]; adaptive personalised federated learning `APFL` proposes a mixture of the local and the global model, where each client trains its local model while contributing to the global model. The personalised model that adapts better to the user’s data emerges from the optimization of a convex combination of the local and global model [8]. These approaches tackle the data heterogeneity problem by making use of a global model to later derive a personalised model, indirectly assuming a common model architecture between all users, thus not taking into account device heterogeneity. This lies in contrast to the proposed approach here, where the model itself is split into “modules” that are federated within their common classes, allowing to have different model architecture and size per client by combining different “modules”. A recent approach that uses clustering between similar users is `PerFed-CKT` [9]. Here the authors propose clustered co-distillation where clients transfer their knowlege to other clients that have similar data-distribution. Because in this setup logits are shared instead of model parameters clients with different model architectures and sizes can federate between each other, thus tackling the device and data heterogeneity problem. This approach uses a common dataset that is shared between all clients, the clustering is based on the similarity of the models output of the common dataset. Similar to the present work, `PerFed-CKT` tackles the device and data heterogeneity problem, but differs in that `PerFed-CKT` does need a common (public) data set shared between all clients, whereas `ModFL` does not need such a public dataset. Another difference is the global model aggregation, `ModFL` averages over the model parameters, whereas `PerFed-CKT` does averages over the logits of the clients models output based on the public dataset.

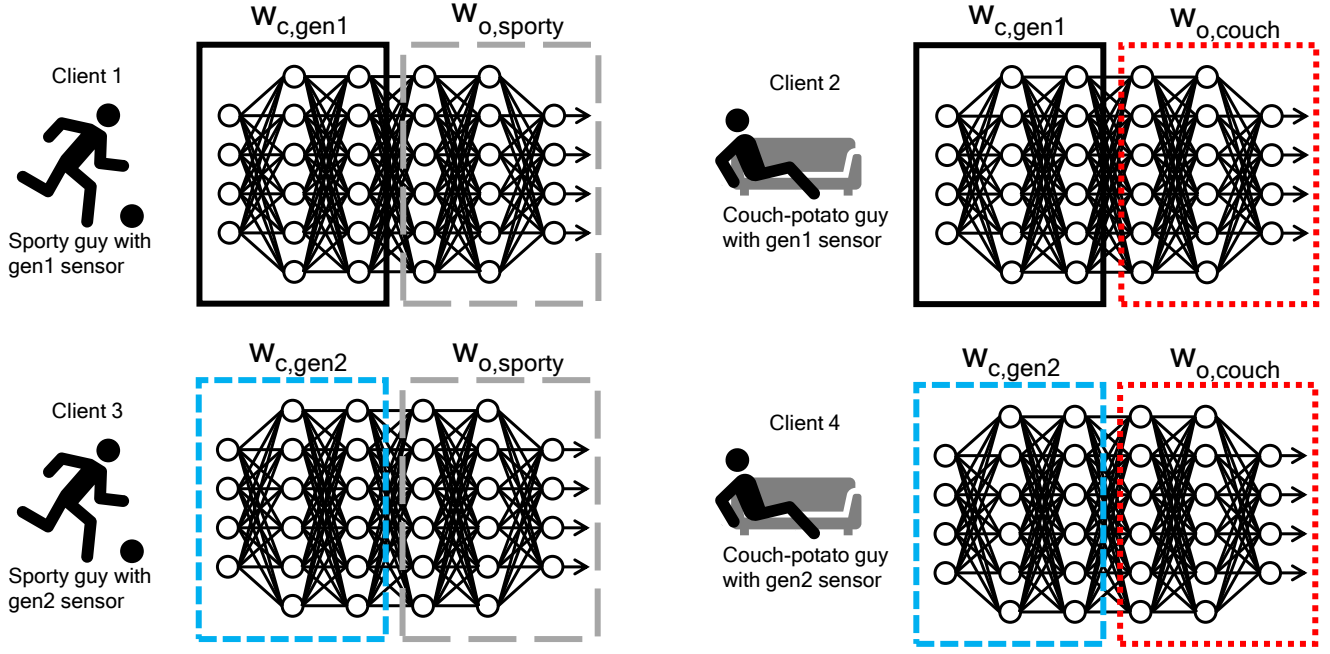


Fig. 1: Main idea of ModFL, where the neural network models are split into a configuration module and an operation module and federated with similar cohorts denoted by the weight notations and boxed with various line styles and colors.

III. MODFL FRAMEWORK

Modular federated learning (ModFL) is inspired by Scania's modular system [10] and by modularity in neural networks [4], [5] combined with federated learning with personalisation layers FedPer [3]. For this ModFL concept, consider first a deep neural network model M_1 with many layers where we divide the model into two modules; a configuration module $W_{c,1}$ consisting of the input layer with some following hidden layers and an operation module $W_{o,1}$ consisting of the rest of the hidden layers and the output layer. Similar to FedPer, the configuration module $W_{c,1}$ will be federated among all clients with model M_1 . Now consider another model M_2 that performs the same task as M_1 but for a different generation of edge device, and similarly divide the model into a configuration module $W_{c,2}$ and a operation module $W_{o,2}$. We have the assumption that different generation edge devices, even though they perform the same task, *i.e.*, they measure the same observable, generate different type of data. Therefore, $W_{c,1}$ and $W_{c,2}$, are in general different in terms of neurons and layers to accommodate the previously discussed differences. However, the operation modules, $W_{o,1}$ and $W_{o,2}$, have to have the same architecture so that they are able to do federated learning between each other, *i.e.* $W_{o,1} = W_{o,2} = W_o$. If all clients would operate in similar ways, *i.e.*, the data is IID in all clients, then all the operation modules W_o from the different clients could federate between each other. However, in general in a federated learning setup, the clients generate non-IID data. To deal with this, the authors of [3] introduced the personalisation layers, which would be local to every client and would not federate. In ModFL, we

assume that client operations can be categorised, whether from a priori knowledge of the edge device usage (operation) or by a similarity measure such as the one proposed by [11]. To simplify the setup, we assume that a priori knowledge of the clients is available. ModFL then proposes to federate the operation modules between clients belonging to the same operation category cohort. This way the configuration modules of ModFL deal with device heterogeneity while the operation modules deal with data heterogeneity.

The main idea of ModFL is to federate the different configuration and operation modules of the deep neural network with its corresponding peers. This means that the configuration module will not necessary federate with the same peers as the operation module. To illustrate it with an example, we assume that we have four clients, each of them will train a local model $M^{(n)} = W_c^{(n)} + W_o^{(n)}$ where client $n \in \{1, 2, 3, 4\}$ and the addition sign (+) represents concatenation. Two types of edge device generations for human activity categorisation are used by the clients; gen1 has only one sensor for categorising the activities and gen2 has two sensors. Therefore, we have two different model architectures $\{M_{gen1}, M_{gen2}\}$. Clients $C_{gen1} = \{1, 2\}$ use gen1 devices and clients $C_{gen2} = \{3, 4\}$ gen2. Then $M^{(n)} = W_{c,gen1}^{(n)} + W_o^{(n)}|_{n=1,2}$ and $M^{(n)} = W_{c,gen2}^{(n)} + W_o^{(n)}|_{n=3,4}$. Additionally, the clients $O_{couch-potato} = \{2, 4\}$ are known to belong to the "couch-potato" cohort and the clients $O_{sporty} = \{1, 3\}$ to the "sporty" cohort. Therefore, the local models are composed of different configuration modules and operation modules:

- $M^{(1)} = W_{c,gen1}^{(1)} + W_{o,sporty}^{(1)}$,
- $M^{(2)} = W_{c,gen1}^{(2)} + W_{o,couch-potato}^{(2)}$

Algorithm 1 ModFL algorithm

The n -th client**Require:** data $D^{(n)}$

- 1: Recieve (W_c, W_o) from server
- 2: $(W_c, W_o) \leftarrow \text{gradient descent}(W_c, W_o, D^{(n)})$
- 3: Send updated (W_c, W_o) to server

The server**Require:** Set of models $\{M_1, \dots, M_m\}$ corresponding to the different client configurations, models split point s .

- 1: **for all** M_i **do**
- 2: $(W_{c,i}, W_o) \leftarrow \text{split}(M_i)$ based on s
- 3: **end for**
- 4: Initialise all $W_{c,i}$ and W_o at random
- 5: Cluster clients into corresponding configuration groups $\{C_1, \dots, C_m\}$ such that there is a one-to-one mapping $\{C_i\} \rightarrow \{W_{c,i}\}$ for $i \in \{1, \dots, m\}$
- 6: Send $(W_{c,i}, W_o)$ to the corresponding clients according to the mapping
- 7: **for** $k = 1, 2, \dots$ **do**
- 8: Receive $W_{c,i}^{(n)}$ and $W_o^{(n)}$ from all clients
- 9: **for** $i \in \{1, \dots, m\}$ **do**
- 10: $W_{c,i} \leftarrow W_{c,i}^{(n)} \quad \forall n \in C_i$
- 11: **end for**
- 12: Cluster clients into operational clusters $\{O_1, \dots, O_l\}$, either by a-priori operational knowledge or using a clustering method on the operational modules $W_o^{(n)}$ and label them $W_{o,j}^{(n)}$ such that there is a one-to-one mapping $\{O_j\} \rightarrow \{W_{o,j}\}$ for $j \in \{1, \dots, l\}$
- 13: **for** $j \in \{1, \dots, l\}$ **do**
- 14: $W_{o,j} \leftarrow W_{o,j}^{(n)} \quad \forall n \in O_j$
- 15: **end for**
- 16: Send the global configuration modules $W_{c,i}$ to the corresponding clients in C_i
- 17: Send the global operational modules $W_{o,j}$ to the corresponding clients in O_j
- 18: **end for**

- $M^{(3)} = W_{c, \text{gen2}}^{(3)} + W_{o, \text{sporty}}^{(3)}$
- $M^{(4)} = W_{c, \text{gen2}}^{(4)} + W_{o, \text{couch-potato}}^{(4)}$

see Fig. 1 for clarity of this example. In this simple scenario the model $M^{(1)}$ will benefit from learning the features of the gen1 edge device from the federation with user $n = 2$, simultaneously model $M^{(1)}$ will benefit to improve the classification performance of the human activities relevant for a sporty user by federating with user $n = 3$. Note that the ModFL concept is not limited to only two configuration groups and operation groups but can be extended to multiple groups.

The detailed and generalized algorithm of ModFL is shown in Algorithm 1. Each client gets weights from the server, updates the weights based on their own data $D^{(n)}$ with a gradient descent algorithm and sends back the updated weights to the server. This continues in many communication rounds until

convergence or requirements are met. The server initialises the models M_i according to the number of different configuration or generations and splits the models to configuration and operation modules $(W_{c,i}, W_o)$. The corresponding weights are sent to the clients with the corresponding configuration group C_i for training. For each step, the server receives updated weights $(W_{c,i}^{(n)}, W_o^{(n)})$ from all clients. For all the different configuration groups $\{C_i\}_{i=1, \dots, m}$, average the weights $W_{c,i}^{(n)}|_{n \in C_i}$ from the clients belonging to that group. Similarly for operation groups $\{O_j\}_{j=1, \dots, l}$ with weights $W_{o,j}^{(n)}|_{n \in O_j}$. Lastly, send all the aggregated modules $\{W_{c,i}\}$ and $\{W_{o,j}\}$ to the corresponding clients in $\{C_i\}$ and $\{O_j\}$, respectively.

ModFL is not limited to a single task problem with one and same operation module W_o . It can be extended to multitask learning problems [12], where the operation module models can be different for different tasks, *i.e.* each task can be represented with different neural network model for the operation module. For example using the same setting in the above example, one task could be to detect injury while sporting for the sporty cohort and another task could be to predict falls for the couch-potato cohort. These two different tasks will in general be different models and will be reflected in the operation module. As a reminder, configuration modules for gen1 and gen2 are in general different models to accommodate the differences of the component generations. The computational complexity for ModFL does not change on the client side compared to FedAvg as it does the same local training. However, on the server side, the computational complexity remains if a-priori operational knowledge is known (row 12 in Algorithm 1), otherwise it may increase depending on the clustering method.

IV. EXPERIMENTAL SETUP

In order to show the concept of ModFL, at least two different datasets with similar tasks are required. We evaluate the performance of ModFL with two different classification tasks, namely image and human activity classification, using CNN and RNN models, respectively. Additionally, the data is non-identical partitioned on the labels among operation groups, thus tuning the degree of data heterogeneity. We will benchmark the results with federated learning with personalisation layer FedPer and the vanilla federated learning FedAvg.

A. Datasets

a) *Image classification:* CIFAR-10¹ [13] and STL-10² [14] are both image classification datasets with 10 class labels with different image resolutions, 32×32 and 96×96 pixels, respectively and each image has a unique label. Both datasets have 9 labels in common, which makes this suitable to test ModFL. Due to the difference in the number of data samples, we evened the amount of data to 11700 data samples per dataset with a 75/25 train and test ratio.

¹<https://www.cs.toronto.edu/~kriz/cifar.html>

²<https://cs.stanford.edu/~acoates/stl10/>

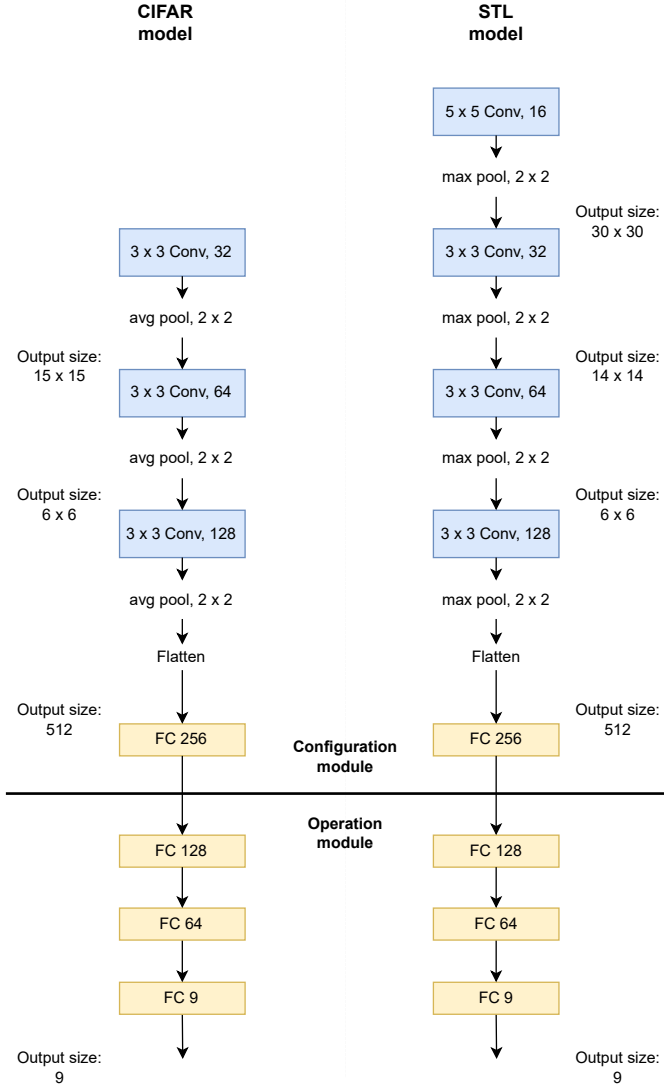


Fig. 2: Image classification models.

b) *Human activity classification*: HAPT³ [15], WISDM⁴ [16], and RealWorld HAR⁵ [17] (RWHAR) are three different human activity recognition datasets, in which they have 5 labels in common. Each data sample has a unique label. WISDM collects accelerometer time-series data from smartphones with a frequency of 20Hz, while HAPT and RWHAR collect accelerometer and gyroscope time-series data with a 50Hz frequency. Although RWHAR has sensors in different locations (head, chest, arm, waist and more), only the one from the waist was used as that is similar to WISDM. Due to the difference in the number of data samples for each label and dataset, we evened the amount of data to 1250 data samples per dataset with a 80/20 train and test ratio.

³<http://archive.ics.uci.edu/ml/datasets/Smartphone-Based+Recognition+of+Human+Activities+and+Postural+Transitions>

⁴<https://www.cis.fordham.edu/wisdm/dataset.php>

⁵https://sensor.informatik.uni-mannheim.de/#dataset_realworld

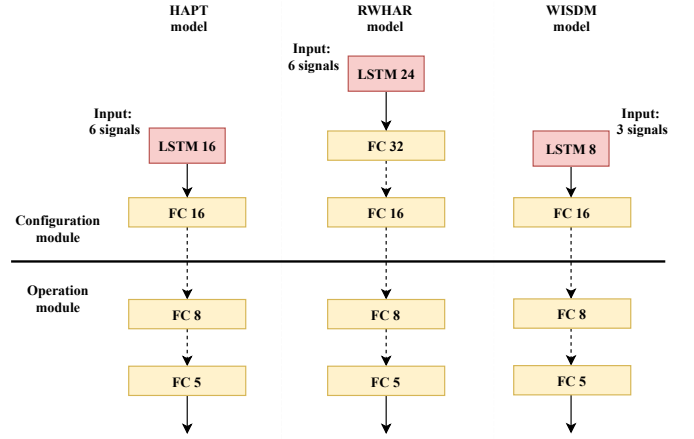


Fig. 3: Human activity classification models.

B. Model Architectures and Hyperparameters

We designed our own models to evaluate ModFL. Recall that the configuration modules can be different to handle different datasets while the operation modules have to be the same for the same task, in this case for classification, across the models. Our aim is not to get the highest test accuracy with these proposed models, but rather show that the ModFL concept can improve it while addressing device and data heterogeneities.

a) *Image classification*: For the image classification, we use CNN layers followed by fully connected layers. Since CIFAR-10 and STL-10 have different resolution images, we designed the configuration module layers differently while the operation module layers remain the same. For the CIFAR model, we designed the configuration module with 32–64–128 units of 2D convolutional layers with average pooling in between followed by a 256 units fully connected layer, in total 4 layers. The configuration module for STL model is designed with 16–32–64–128 units of 2D convolutional layers with max pooling in between followed by a 256 units fully connected layer, in total 5 layers. The operation module for both models are 128–64–9 units of fully connected layers. See Fig. 2 for an illustration of the models. All layers used ReLU (rectified linear unit) as activation function except the last fully connected layer which used softmax.

b) *Human activity classification*: Since the human activity classification datasets are time-series data, the natural way for classification is to use RNN, for this case we used long-short term memory (LSTM), with fully connected layers. For the HAPT model, we designed the configuration module with a 16 units LSTM layer followed by a 16 units fully connected layer. For the RWHAR model, we used a 24 units LSTM layer with 32–16 units of fully connected layers. Lastly, for the WISDM model we had an 8 units LSTM layer with a 16 units fully connected layer. The operation module consisted of 16–5 units of fully connected layers. See Fig. 3 for an illustration of the models. All layers used tanh (hyperbolic

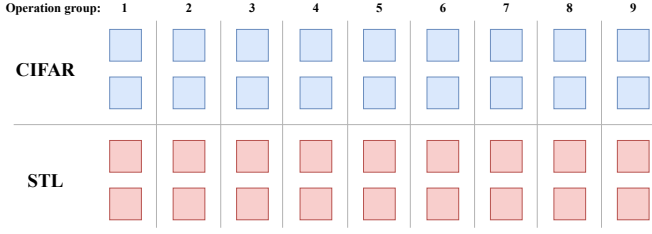


Fig. 4: Example of client setup with $N = 36$ clients, where 18 clients use CIFAR model and the other 18 clients the STL model (*i.e.* $|\{C_i\}| = 2$) and $|\{O_j\}| = 9$ different operation groups with 4 clients in each group.

tangent) as activation function except the last fully connected layer which used softmax.

For the federated learning training to converge easier, the weight initialisation should be the same for all clients [1]. In our case, the configuration module layers from the same model have to be initiated with the same weights and the operation module layers for all clients have to be initiated with the same weights. Similarly to the conclusions from [3] where different models have slightly better test accuracy with different amount of personalised layers, we conclude that the cut between the configuration module and the operation module of a model is an additional hyperparameter to be tuned for each specific case.

We use Adam optimiser with a learning rate of 0.001. For training the image classification, we set local epochs to 1, the global communication rounds to 200, and a batch size of 16. For training the human activity classification, we set local epochs to 1, the global communication rounds to 2000, and a batch size of 8. The server receives all the incoming weights, divides them into configuration and operation modules, and computes federated averaging FedAvg [1] accordingly to the groups.

C. Client Setup

In our setup, every client has the same amount of data and the same amount of data per given label. For the image classification task the number of clients is $N \in \{18, 36, 54, 72\}$, half of them use the CIFAR-10 dataset and model and the other half use the STL-10 dataset and model, *i.e.* $|\{C_i\}| = 2$. We divide the the total number of clients into $|\{O_j\}| = 9$ different operation groups, *i.e.*, if $N = 36$, then 18 clients use the CIFAR model and the other 18 the STL model. Additionally, these 36 clients are divided into 9 groups (4 clients in each group), which will represent different operations, see Fig. 4 for clarity. Note that since we have limited amount of data for both tasks, the more clients we have for training, the less data each client has.

For the human activity classification task, the number of clients is $N \in \{15, 30, 45, 60\}$. Since there three different datasets are used, $|\{C_i\}| = 3$, a third use HAPT model, another third the RWHR model and last third the WISDM

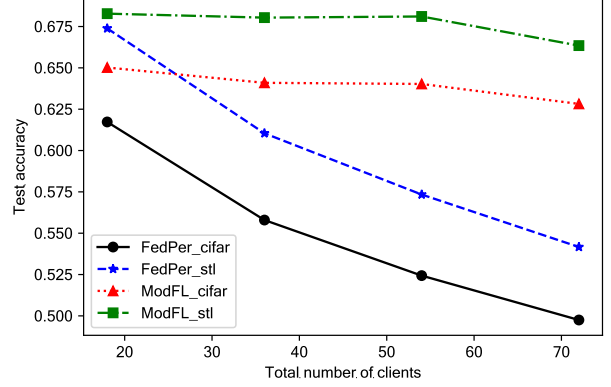


Fig. 5: Test accuracy with respect to the number of clients on the image classification task with $P_{img} = 6$. ModFL does not suffer as much compared to FedPer as the number of clients increases and the amount of data per client decreases.

model. We divide the total number of clients into $|\{O_j\}| = 5$ different operation groups.

Furthermore, we create three different levels of data heterogeneities P . For the image classification task, we chose $P_{img} \in \{3, 6, 9\}$ unique labels per operation group and for the human activity classification task we chose $P_{har} \in \{2, 4, 5\}$, *e.g.*, $P_{img} = 3$ means that each operation group has a set of 3 labels of unique combination. However, note that some labels will overlap with other operation groups. $P_{img} = 9$ corresponds to all labels, which is an IID case, and this means that all clients belong to one and the same operation group ($|\{O_j\}| = 1$), similarly when $P_{har} = 5$. This reduces the problem to FedAvg on the operation module for all the clients. This special case can also be seen as kind of reversed FedPer, the configuration module is personalised to the generation of devices and the operation module is federated among all the clients.

Note that when we compare ModFL to FedPer or FedAvg, we split the ModFL results into the respective dataset models (*e.g.* CIFAR and STL models). This way, we can compare the differences since FedPer and FedAvg needs to have the experiments separated (*e.g.* one run with CIFAR model and dataset and another run with STL model and dataset).

V. RESULTS

A. Image Classification with CNN

To study the potential improvement of performance of ModFL over other federated learning frameworks, we run different number of clients to federate among. With a finite dataset divided into different clients, a large number of clients N results in less data per client to train the local models. For the FedPer framework this translates into less data for training the personalisation layers. In Fig. 5 (averaged across same dataset clients), we see that as N increases, performance

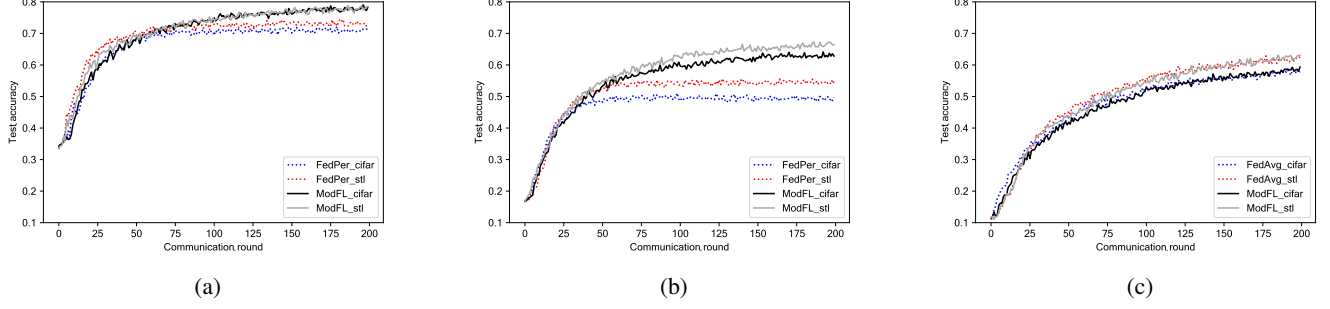


Fig. 6: Performance of ModFL on image classification using CNN: (a) Non-IID partition with $P_{img} = 3$ labels compared to FedPer, (b) Non-IID partition with $P_{img} = 6$ labels compared to FedPer, (c) IID partition compared to FedAvg.

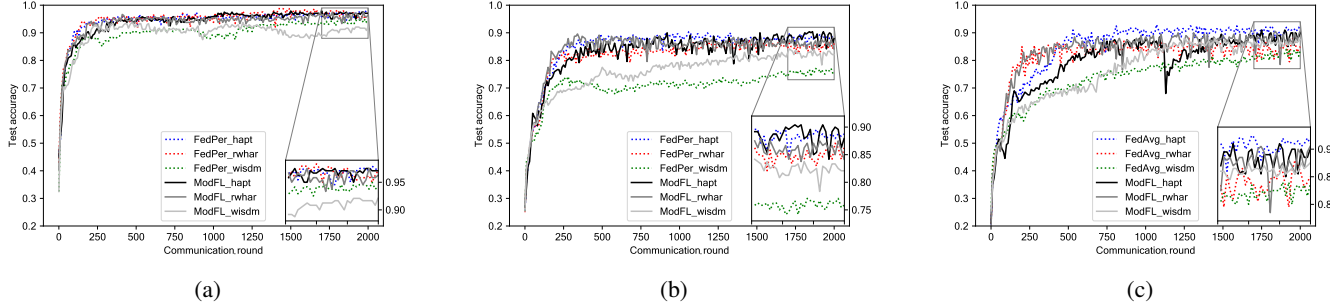


Fig. 7: Performance of ModFL on human activity classification using RNN (the last 300 rounds are zoomed in): (a) Non-IID partition with $P_{har} = 2$ labels compared to FedPer, (b) Non-IID partition with $P_{har} = 4$ labels compared to FedPer, (c) IID partition compared to FedAvg.

TABLE I: Test accuracy on image classification. Note that FedX is FedPer for non-IID cases and FedAvg for IID case.

	P_{img}	3	6	9
Models	(non-IID)	(non-IID)	(IID)	
FedX_cifar		71.23%	49.75%	57.92%
ModFL_cifar		78.29%	62.82%	59.36%
FedX_stl		72.09%	54.17%	62.59%
ModFL_stl		78.46%	66.35%	63.07%

of FedPer decreases. In contrast, the performance of ModFL suffers marginally as N increases. This is due to the fact that ModFL takes advantage from the federation of *like-minded* clients belonging to the same operation group.

For all tested number of clients N with $P_{img} = \{3, 6\}$ labels, we find that ModFL performs better than FedPer for the CIFAR and the STL dataset. Fig. 6 shows the test accuracy (averaged across same dataset clients) as a function of the number of communication rounds for $N = 72$ clients and different data heterogeneities. In Fig. 6a the number of operation groups is $|\{O_j\}| = 9$ with $P_{img} = 3$, here the accuracy of FedPer for CIFAR and STL starts to stagnate after about 100 communication rounds, contrary ModFL continuously improves the accuracy, beyond FedPer, and does not seem to stagnate even after 200 rounds. In Fig. 6b the number of operation groups is $|\{O_j\}| = 9$ with $P_{img} = 6$ and again the accuracy of FedPer stagnates but at around

50 communication rounds. The ModFL accuracy continuously increases even after 200 rounds. Do note that the total number of models of FedPer is equal to the number of clients and in the case of ModFL the total number of models is 18, *i.e.*, equal to the total number of possible combinations of configuration modules and operation modules. Fig. 6c where $P_{img} = 9$, we compare ModFL to the *vanilla* FedAvg due to this case being IID where FedAvg performs better than FedPer. We see that in the IID data case ModFL performs as good as FedAvg, thus the proposed protocol implementation deals IID data as good as the FedAvg approach with federated learning. The accuracy results are summarised in TABLE I.

B. Human Activity Classification with RNN

We show results of the ModFL framework applied to time-series data by evaluating the accuracy of human activity classification using an RNN model. Fig. 7a shows the test accuracy (averaged across same dataset clients) as a function of the communication rounds for the FedPer and the ModFL frameworks evaluated for the HAPT, RWHR and WISDM datasets with a number of different operation groups of $|\{O_j\}| = 5$ and level of data heterogeneity $P_{har} = 2$. Here the accuracy between the two frameworks is basically the same for the HAPT and RWHR models. For the WISDM model, the accuracies for both frameworks do not overlap as the accuracy for both frameworks fluctuates. Fig. 7b shows the test accuracy

TABLE II: Test accuracy on human activity classification. Note that FedX is FedPer for non-IID cases and FedAvg for IID case.

Models \ P_{img}	2 (non-IID)	4 (non-IID)	5 (IID)
FedX_hapt	97.08%	88.75%	91.00%
ModFL_hapt	97.08%	87.92%	90.00%
FedX_rwhar	97.08%	86.25%	85.00%
ModFL_rwhar	96.25%	84.17%	90.50%
FedX_wisdm	94.17%	75.83%	83.50%
ModFL_wisdm	92.08%	82.08%	87.50%

with a number of different operation groups $|\{O_j\}| = 5$ and level of data heterogeneity $P_{har} = 4$. The accuracy curves for both frameworks overlap for the HAPT and RWHAR datasets and for the WISDM dataset the curves fluctuate. No clear tendency from either of the frameworks can be drawn from this data. We get similar results for the experiments with all different N client numbers. Fig. 7c shows the test accuracy only one operation group $|\{O_j\}| = 1$ containing all 5 labels, *i.e.*, IID data. We observe that for RWHAR, the curves fall into each other, but for the HAPT and the WISDM the accuracy curves fluctuate and do not fall onto each other. We attribute the fluctuations in the data to sensitivity in initial conditions. The accuracy results are summarised in TABLE II.

Comparing the results in Fig. 7 with the ones in the other experiments with different amount of N clients (not shown), we conclude that the fluctuations are inherent to the training on the datasets and no conclusion can be drawn about on which framework is preferably used in with this dataset. What the data suggests is that ModFL for time-series data is as good as FedPer for non-IID data and FedAvg for IID data. Moreover, we argue that the lack of the performance improvement of ModFL over FedPer is due to the nature of the dataset; the accuracy of the models for all three frameworks is relatively high ≈ 0.8 or higher, even for the vanilla FedAvg, leaving little room for performance improvement. We think that to exploit the potential of ModFL, more challenging datasets than the chosen human activity classification are needed.

VI. CONCLUSIONS

In this work, we propose ModFL a novel framework to train neural networks in a federated learning fashion when non-IID data is generated and the edge devices are heterogeneous, *i.e.*, they belong to different generations and thus have different kind of data but perform the same task. We achieve this by extending the FedPer approach of splitting the neural networks into configuration modules (layers) and operation modules (layers) and let the configuration modules federate between same kind of devices while like-minded users are allowed to federate the operation modules. Results on CIFAR-10 and STL-10 data indicate that the ModFL protocol for CNN is well suited to take full advantage of the a federated

learning approach among different groups of peers and show that it outperforms FedPer. It serves as an alternative to FedPer when the individual clients do not have sufficient data to train the personalisation layers. The results on the time-series data using RNN are inconclusive and further research on more challenging datasets is needed. Further work on more challenging time-series data is desired to understand better the advantages and limitations of the proposed framework in this kind of data.

VII. ACKNOWLEDGMENTS

We thank Anders Vesterberg for insights and useful discussions. This work was supported by VINNOVA within the FFI program under contract 2020-02916.

REFERENCES

- [1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 54, 2017. [Online]. Available: <https://proceedings.mlr.press/v54/mcmahan17a.html>
- [2] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.
- [3] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary, "Federated learning with personalization layers," dec 2019. [Online]. Available: <http://arxiv.org/abs/1912.00818>
- [4] C. Watanabe, K. Hiramatsu, and K. Kashino, "Modular representation of layered neural networks," *Neural Networks*, vol. 97, pp. 62–73, mar 2017.
- [5] D. Filan, S. Casper, S. Hod, C. Wild, A. Critch, and S. Russell, "Clusterability in neural networks," 2021. [Online]. Available: <http://arxiv.org/abs/2103.03386>
- [6] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proceedings of Machine Learning and Systems*, vol. 2, 2020, pp. 429–450.
- [7] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning: A meta-learning approach," 2020. [Online]. Available: <http://arxiv.org/abs/2002.07948>
- [8] Y. Deng, M. M. Kamani, and M. Mahdavi, "Adaptive personalized federated learning," 2020. [Online]. Available: <http://arxiv.org/abs/2003.13461>
- [9] Y. J. Cho, J. Wang, T. Chiruvolu, and G. Joshi, "Personalized federated learning for heterogeneous clients with clustered knowledge transfer," 2021. [Online]. Available: <http://arxiv.org/abs/2109.08119>
- [10] M. Sköld, *Modularization: the art of making more by using less*. Rheologica Publishing, 2017. [Online]. Available: <http://modularization.com/>
- [11] F. Sattler, K.-R. Müller, and W. Samek, "Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 8, pp. 3710–3722, 2021.
- [12] M. Crawshaw, "Multi-task learning with deep neural networks: A survey," 2020. [Online]. Available: <http://arxiv.org/abs/2009.09796>
- [13] A. Krizhevsky, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009. [Online]. Available: <https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf>
- [14] A. Coates, H. Lee, and A. Y. Ng, "An analysis of single-layer networks in unsupervised feature learning," in *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 15, 2011. [Online]. Available: <https://proceedings.mlr.press/v15/coates11a.html>
- [15] J. L. Reyes-Ortiz, L. Oneto, A. Samà, X. Parra, and D. Anguita, "Transition-aware human activity recognition using smartphones," *Neurocomputing*, vol. 171, pp. 754–767, 2016. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0925231215010930>
- [16] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," *ACM SIGKDD Explorations Newsletter*, vol. 12, no. 2, pp. 74–82, 2011.

- [17] T. Sztyley and H. Stuckenschmidt, "On-body localization of wearable devices: An investigation of position-aware activity recognition," in *2016 IEEE International Conference on Pervasive Computing and Communications PerCom*. IEEE, 2016, pp. 1–9.