FEDOPENHAR: FEDERATED MULTI-TASK TRANSFER LEARNING FOR SENSOR-BASED HUMAN ACTIVITY RECOGNITION

by

Egemen İşgüder

B.S., Computer Engineering, Galatasaray University, 2019

B.S., Industrial Engineering, Galatasaray University, 2020

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering

Boğaziçi University

2024

ACKNOWLEDGEMENTS

I express my sincere gratitude to my (co-)advisor, Özlem Durmaz İncel, for a multitude of reasons, extending beyond the academic realm. During the pivotal period as a thesis student, she welcomed, assisted, tolerated, and believed in me through the most challenging times. It is an honour to acknowledge her role as my guardian angel advisor.

My heartfelt thanks extend to the members of my jury: Atay Özgövde, a constant presence since the start of my academic journey; İnci Baytaş, to whom I owe my machine learning knowledge and passion, as well as the pathway to my first academic publication; and Özgün Pınarer, an enduring source of support during my time at Galatasaray University and a continuing role model as an engineer-academic.

I would like to thank Cem Ersoy for accepting me into NetLAB and providing solace even during challenging events, making Boğaziçi feel like a safe haven. Special thanks are also owed to Murat Özyurt and Birkan Yılmaz for their unwavering academic and emotional support.

I extend my thanks to my family, my parents for a lifetime of support, and my chosen family - Aytekin Şatıroğlu, Onur Kılınç, and Kıvanç Komesli - for bringing genuine friendship into my life.

I would also like to thank Elif Aksoy, for helping me by any means possible, and for her guidance and wisdom.

I would like to express my gratitude to ASUS for creating the most amazing device that helped me disperse the boredom of the academy, ROG Ally.

Lastly, it is important to mention that this thesis is not funded by any person, corporation or foundation.

Dedicated to Mustafa Kemal Atatürk.

The supreme guide in life is science.

ABSTRACT

FEDOPENHAR: FEDERATED MULTI-TASK TRANSFER LEARNING FOR SENSOR-BASED HUMAN ACTIVITY RECOGNITION

Motion sensors integrated into wearable and mobile devices provide valuable information about the device users. Machine learning and, recently, deep learning techniques have been used to characterize sensor data. Mainly, a single task, such as the recognition of human activities, is targeted, and the data is processed centrally at a server or in the cloud. However, the same sensor data can be utilized for multiple tasks and distributed machine-learning techniques can be used without the requirement of transmitting data to a centre. In this thesis, we introduce the FedOpenHAR framework that explores *Federated Transfer Learning in a Multi-Task* manner for both sensor-based human activity recognition and device position identification tasks. It utilizes a transfer learning approach by training task-specific and personalized layers in a federated manner. The OpenHAR framework, which contains ten smaller datasets, is used to train the models. The challenge is obtaining robust model(s) applicable for both tasks in different datasets, which may include only some label types. Multiple experiments are carried out in the Flower federated learning environment using DeepConvLSTM, MLP and CNN architectures. Results are presented for federated and centralized versions under different parameters and restrictions. By utilizing transfer learning and training a task-specific and personalized federated model, we obtained a higher accuracy (72.4%) than a fully centralized training approach (64.5%) and a similar accuracy in a case where each client performs individual training in isolation (72.6%). However, the advantage of FedOpenHAR over individual training is that if a new client joins with a new label type (meaning a new task), it can start training from the common layer. Additionally, if a new client wants to classify a new class type in one of the existing tasks, FedOpenHAR can start training directly from task-specific layers. We also perform further experiments to evaluate the performance of FedOpenHAR under different settings.

ÖZET

FEDOPENHAR: SENSÖR-TABANLI İNSAN AKTİVİTESİ TANIMLAMA İÇİN FEDERE ÇOK-HEDEFLİ ÖĞRENME AKTARIMI

Giyilebilir ve mobil cihazlara entegre edilmiş hareket sensörleri, cihaz kullanıcıları hakkında değerli bilgiler sağlar. Makine öğrenimi ve son zamanlarda derin öğrenme teknikleri, sensör verilerini karakterize etmek için kullanılmıştır. Genellikle, insan aktivitelerinin tanınması gibi tek bir görev hedeflenir ve veri genellikle bir sunucuda veya bulutta merkezi olarak işlenir. Ancak aynı sensör verisi birden fazla görev için kullanılabilir ve verilerin merkeze iletilmesi gereksinimi olmaksızın dağıtılmış makine öğrenimi teknikleri kullanılabilir. Bu tezde, hem sensör tabanlı insan aktivitesi tanıma hem de cihaz konumu belirleme görevleri için Çok-Hedefli Federe Öğrenme Aktarımı'nı kullanan FedOpenHAR metodunu sunuyoruz. Hedefe-özel ve kişiselleştirilmiş katmanları federe bir şekilde eğiterek öğrenme aktarımı yaklaşımını kullanıyoruz. Modelleri eğitmek için on küçük veri seti içeren OpenHAR verisetleri kümesi kullanılıyor. Temel amaç ve zorluk, farklı veri kümelerinin yalnızca bazı etiket türlerini içerirken, farklı veri kümelerinde her iki göreve de uygulanabilir sağlam model(ler) elde etmektir. Çeşitli parametreler ve kısıtlamalar altında, Flower federe öğrenme ortamında DeepConvLSTM, MLP ve CNN mimarileri kullanılarak birden fazla deney gerçekleştirmektedir. Sonuçlar, federe ve merkezi deneyler için sunulur. Öğrenme aktarımı kullanarak ve hedefe-özel ve kişiselleştirilmiş federe bir model eğiterek, tamamen merkezi bir eğitim yaklaşımından (64.5%) daha yüksek bir doğruluk elde ettik (72.4%) ve her istemcinin izole bir şekilde bireysel eğitim yaptığı bir durumla benzer bir doğruluk elde ettik (72.6%) . Ancak, FedOpenHAR'ın bireysel eğitime göre avantajı, yeni bir istemcinin yeni bir etiket türüyle (yeni bir hedef) katılması durumunda ortak katmandan eğitime başlayabilmesidir. Ayrıca, yeni bir istemcinin mevcut görevlerden birinde yeni bir sınıf türü sınıflandırmak istemesi durumunda, FedOpenHAR doğrudan hedefe-özel katmanlardan eğitime başlayabilir. Ayrıca FedOpenHAR'ın farklı parametreler altındaki performansını değerlendirmek için daha fazla deney yapılmış ve sonuçları sunulmaktadır.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS . . . iii
ABSTRACT . . . v
ÖZET . . . vi
LIST OF FIGURES . . . ix
LIST OF TABLES . . . xi
LIST OF SYMBOLS . . . xii
LIST OF ACRONYMS/ABBREVIATIONS . . . xiii
1. INTRODUCTION . . . 1
2. RELATED WORK . . . 4
3. DATASETS . . . 9
3.1. Properties of OpenHAR . . . 9
3.2. Preprocessing . . . 11
3.2.1. Activities . . . 11
3.2.2. Positions . . . 12
3.2.3. Users . . . 13
4. FEDOPENHAR METHOD . . . 20
4.1. Federated Learning . . . 20
4.1.1. Federated Aggregation Algorithms . . . 21
4.1.1.1. FedAVG . . . 21
4.1.1.2. FedProx . . . 21
4.1.1.3. FedYogi . . . 22
4.1.2. Federated Simulation Framework . . . 22
4.2. FedOpenHAR . . . 23
4.3. Models for Training . . . 25
4.3.1. MLP . . . 25
4.3.2. CNN . . . 25
4.3.3. DeepConvLSTM . . . 26
4.3.4. Metrics of Success . . . 27

5. INITIAL EXPERIMENTS 31
5.1. Centralized Experiments 31
5.2. Individual Experiments 32
5.3. Federated Experiments 33
5.3.1. Federated One-Task Experiments 33
5.3.2. Federated Multi-Task Experiments 34
5.3.3. FedOpenHAR 35
6. FURTHER EVALUATION OF FEDOPENHAR 44
6.1. Impact of the Number of Pretraining Epochs 44
6.2. Different Number of Personalization Epochs 45
6.3. Optimizing All Layers vs. Only the Last Layer After Transfer Learning 45
6.4. Changing the Federated Averaging Algorithm 46
6.5. Changing the Layer Sizes in the Federated Hierarchy 46
6.6. Using Various Training Percentages 47
6.7. Using FedOpenHAR in a Single Dataset 48
6.8. Different Ratio of Clients Sampled at Training 49
7. CONCLUSION 56
REFERENCES 58

LIST OF FIGURES

Figure 3.1. Activity label distribution of OpenHAR, unprocessed. 14

Figure 3.2. Activity label distribution of OpenHAR, after preprocessing. . . . 14

Figure 3.3. Position label distribution of OpenHAR, unprocessed. 15

Figure 3.4. Activities of OpenHAR, after preprocessing, each dataset. 16

Figure 3.5. Position label distribution of OpenHAR, after preprocessing. . . . 17

Figure 3.6. Positions of OpenHAR, after preprocessing, each dataset. 18

Figure 3.7. User distribution of OpenHAR, for each dataset. 19

Figure 4.1. Algorithm of FedAVG. 29

Figure 4.2. Layered hierarchy of FedOpenHAR. 29

Figure 4.3. Algortihmic flow of FedOpenHAR. 30

Figure 5.1. Centralized activity accuracies. 37

Figure 5.2. Centralized position accuracies. 37

Figure 5.3. Individual accuracies for 9. dataset. 38

Figure 5.4. Client-wise comparison of accuracies of individual clients. 38

Figure 5.5. Federated one-task activities' accuracies. 39

Figure 5.6. Federated one-task positions' accuracies. 39

Figure 5.7. Federated multi-task accuracies. 40

Figure 5.8. Comparison of weighted average accuracies of clients by layers. . . 40

Figure 5.9. Client-wise comparison of accuracies. 41

Figure 5.10. Client-wise & layerwise comparison of FedOpenHAR. 41

Figure 5.11. Weighted average accuracies of clients by layers. 42

Figure 6.1. Weighted average accuracies of different training percentages. . . . 50

Figure 6.2. Weighted average accuracies of user-wise division in dataset 9. . . 50

Figure 6.3. Activity distribution for userwise division of dataset 9. 51

Figure 6.4. Position distribution for userwise division of dataset 9. 52

Figure 6.5. Weighted average accuracies of client selection percentages. 53

LIST OF TABLES

Table 2.1. Comparison with related studies. 8

Table 3.1. Details of 10 datasets of OpenHAR. 10

Table 5.1. Different centralized approaches' accuracies. 42

Table 5.2. Individual experiments' accuracy results. 42

Table 5.3. Weighted accuracies for federated experiments. 43

Table 6.1. Weighted accuracies for different pretraining epochs. 53

Table 6.2. Weighted accuracies for different personalization epochs. 54

Table 6.3. Weighted accuracies for different optimizing approaches. 54

Table 6.4. Weighted accuracies for various federated averaging algorithms. . . 54

Table 6.5. Weighted accuracies for various layer sizes. 55

Table 6.6. Weighted accuracies for various division types of dataset 9. 55

LIST OF SYMBOLS

β_1	Momentum parameter in FedYogi
β_2	Second moment parameter in FedYogi
η	Server-side learning rate in FedYogi
η_l	Client-side learning rate in FedYogi
μ	Regularization coefficient in FedProx
τ	Controls the algorithm's degree of adaptability in FedYogi

LIST OF ACRONYMS/ABBREVIATIONS

CNN	Convolutional Neural Network
FC	Fully Connected
FL	Federated Learning
GDPR	General Data Protection Regulation
HAR	Human Activity Recognition
KNN	K-Nearest Neighbor
LSTM	Long Short-Term Memory
MLP	Multi Layer Perceptron
ReLU	Rectified Linear Unit
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine

1. INTRODUCTION

Motion sensors available on wearable and mobile devices are commonly used to characterize the habitual patterns of their users: activities performed, where they are, and where they carry their devices. Particularly, monitoring people's physical activity levels can help them stay active and reduce their risk of chronic diseases, such as obesity. In the literature, machine learning and, recently, deep learning techniques are applied to the sensor data [1]. Although there are efforts to train and run the learning algorithms on the device, the wearables are still resource-limited regarding computation and memory, and processing is performed at a central server or a cloud environment. Wearable and mobile devices are personal devices, and the integrated sensors collect privacy-sensitive personal data. Central collection and processing of such data may violate the users' privacy. In this respect, federated learning is an emerging approach [2].

Federated learning is a novel technique in machine learning that was first announced by Google [3]. The main goal of federated learning is processing data while preserving privacy. Data privacy has become a significant national and international concern, especially with recent updates on the General Data Protection Regulation (GDPR). Federated learning offers a critical solution at this point. Instead of sharing data among learning participants (clients), model parameters are shared, and the learning is realized in a distributed manner. This way, data privacy is assured, and the wireless communication costs decrease because only model parameters are shared between participants instead of big chunks of data. This situation also benefits the smaller devices with resource constraints like wearable devices, smartphones and IoT devices.

Federated learning is applied in various fields, such as healthcare, finance, image, and text processing. It primarily benefits the apps based on machine learning in mobile and wearable devices. An area where federated learning is utilized is sensor-based

human activity recognition (HAR). Data from motion sensors such as accelerometers and gyroscopes are processed with machine learning techniques to monitor a user's activities (walking, sitting, etc.) or special activities such as sports exercises or activities that may include life-threatening situations (e.g. falling). In addition, elderly care and physical rehabilitation activities can be monitored.

In this thesis, for the recognition of human activities and sensor device positions, DeepConvLSTM [4] is applied for centralized and federated learning, both with various methods and challenges. For the learning part, the OpenHAR framework [5] is used, which consists of ten smaller datasets. Combined, it contains accelerometer sensor data positioned in various parts of the body, collected from 211 participants. A different client in the federated learning experiments owns every dataset. In contrast, two versions exist in centralized learning: one similar to the federated setup, in which each dataset is used for centralized learning separately at each client, and another form where all data is combined in a big pile for centralized learning. Federated experiments are also categorized as one-task, multi-task and multi-task with a layered hierarchy. The tasks to learn during the experiments are the users' activities and the devices' positions. In the OpenHAR dataset, the devices were not located in a specific position. To the best of our knowledge, this study is the first to implement a multi-task classification problem in multiple different datasets with a federated transfer learning method, where all datasets may not contain every type of label.

For the federated learning part, the layered hierarchy, named FedOpenHAR, learning consists of pre-trained, common, task-specific and personalized layers, inspired by the example of [6], where they focus on multiple image recognition tasks. One of the clients is selected to pre-train the initial weights, which constitutes the pre-trained layer, and then all clients, without any task restriction, participate in one federated loop, which will create the common layer. Afterwards, there are two federated loops (one for each task) to implement the task-specific layers. Ultimately, every client uses its own data for simple training to obtain the personalized layer. Parameters are combined with the FedAVG algorithm [3] during the federated experiments. In the end,

a model with an accuracy of 72.4% is obtained (which is 0.2% less than the baseline), which works on multiple tasks, 14 different labels, and opens the path to learning new labels and tasks with its layered hierarchy.

Regarding similar works in the field, although federated multi-task learning was utilized for different tasks in the image recognition domain or federated transfer learning in the human activity recognition domain, this thesis is the first to implement human activity and device position recognition with a federated multi-task transfer learning architecture.

The thesis continues as follows: Section 2 examines the recent works in the area, Section 3 explains the used dataset in experiments, while Section 4 presents the selected classification algorithms and the federated learning architecture. Section 5 focuses on the results of initial experiments realized in different conditions with various parameters. After the initial experiments, further advancements of the main proposed method (FedOpenHAR) are presented in Section 6. Section 7 summarizes the results and opens the path for further studies.

2. RELATED WORK

Federated learning has been quite popular in the research community since its introduction in 2017 by Google [3]. The paper by Google offers a gateway into the federated learning domain. In this paper, the authors define the concept of federated learning and introduce the first federated averaging algorithm, FedAVG. Being quite a technical study, they also conducted experiments and showed their results on the CIFAR-10 dataset.

Authors in [7] evaluate the development and evolution of federated learning. Both advantages and disadvantages of the federated learning method, its strengths and weaknesses, are explained in depth. The authors explore the different aspects of existing federated learning studies, such as resource allocation, communication and security. They offer the exploration of challenges, such as dropped participants, unlabeled data, and device heterogeneity, which can be critical points in further studies of federated learning as future research directions.

The study in [8] provides a detailed technical overview of implementing federated learning, demonstrating how it can be integrated with various machine learning approaches. The paper distinguishes and implements horizontal federated learning (where data features are consistent) and vertical federated learning (where data sources are identical), providing insights into these techniques. However, the paper's exploration of federated transfer learning (where data sources and features both differ) remains an open avenue for future research.

In this thesis, our main application area is sensor-based human activity recognition (HAR), and there are different studies in the literature focusing on HAR. As federated learning contains an aggregation of model parameters, choosing an averaging algorithm is important for success. In [9], different aggregation algorithms such as FedPer, FedMA and FedAVG are examined with HAR data and their results are

compared. They found that the baseline FedAVG algorithm performed better than the other algorithms. As FedMA and FedPer have additional designs to improve the personalization of local models, their averaging of more personalized models leads to a negative effect that reflects the lower accuracy when evaluated on the global test set.

One of the pioneering works on federated learning for sensor-based HAR is [10], which proposes a federated learning platform for wearable healthcare devices. The study employs the UCI public dataset for smartphones [11] for training models. This dataset is a subset of the OpenHAR-integrated datasets utilized in this research. FedHealth, [10] employs machine learning techniques like K-nearest neighbours (KNN) and support vector machines (SVM) instead of the deep learning approaches utilized in this thesis.

Some studies utilizing federated learning in HAR compare the centralized approach vs federated learning. As in [12], authors compare centralized deep learning with federated learning for classification activities. They achieve similar success with both techniques regarding macro F1-scores and confusion matrices. Federated learning, which can be as effective as centralized learning, has the privacy advantage.

Clients (participants) of the federated learning method differ in technical capabilities, data contents and features, which is called heterogeneity. This deeply affects the outcome of the federated learning method. Personalizing the obtained model for every client can be a useful solution. Personalization in federated learning has been previously investigated in [13] through a new federation algorithm called FedPer. This algorithm personalizes the last layer of models across all clients using a specialized federated averaging algorithm, incorporating additional computing steps involving stochastic gradient descent. Personalization is also studied in [14, 15]. Authors in [14] conclude that FedAVG may not work with severely skewed data, and develop a personalized FedAVG. In the study of [15], transfer learning is used for personalization, and FedMD with FedStag is used as a federated aggregation algorithm.

Federated learning is generally applied to structured learning, defined as learning in distinct steps such as training and testing. However, authors in [16] offer a new approach to federated learning, which combines active learning and label propagation. In this study, active learning is realized by asking the users about the labels, which are then propagated to unlabeled data.

During the training part of the clients, an important decision to take is to determine which clients will participate in the following training round. Different studies exist about this decision; for example, in [17], authors implement federated learning in the HAR area where they drop the slow or little co-related nodes. Another study is [18], where the authors focus on clustering to select model weights.

Privacy is preserved in federated learning by not sharing private data among the participants, but security is not guaranteed. Some malicious participants can misdirect the model's aggregation or obtain information about other participants. In [19], authors introduce a new form of inference-based privacy attacks called Time Inference Attacks, which can reveal private time-related properties such as the presence or absence of a sensitive feature over time and if it is periodical.

Combining multi-task learning with federated learning has been quite a recent approach; one of its examples can be seen in [20]. The authors introduce a framework called MOCHA, which enables simultaneous optimization of linear and nonlinear learning tasks. Their experiments utilize three diverse datasets, including a human activity recognition dataset where each individual is treated as a different task.

Another work that explores federated learning and meta-learning is [21], which proposes a meta-learning approach to federated HAR. The authors utilize both sensor and survey data for training, employing convolutional neural networks (CNNs) as the primary method. This approach involves clustering individuals and selecting temporal features to personalize the global model obtained through federated learning.

One of the most sophisticated studies in federated learning, encompassing multiple aspects such as multi-task learning, layered hierarchy, and transfer learning, is presented in [6]. The authors propose a novel federated learning architecture combining multi-task, transfer and federated learning. Their proposed algorithm is structured into four distinct layers: pre-trained, common, task-specific, and personalized layers. Their experiments use an artificially generated network consisting of face images. The dataset labels are binary, containing only two distinct categories. They achieve comparable results to their baseline (individually trained) method with their proposed approach. A key distinction between our study and theirs lies in the data distribution. Rather than dividing a single dataset into separate clients, each client utilizes an entirely distinct dataset, which may or may not include all the available labels. Additionally, they target image recognition tasks, whereas our focus is on sensor-based HAR and the baseline deep learning architectures are different. In addition, in [6], the labels to classify are binary for each task, whereas in the thesis, the labels are not binary but multiple.

To date, federated multi-task learning has been employed for various tasks in the image recognition domain, and federated transfer learning has been applied in the human activity recognition domain. However, this thesis is the first study on sensor-based HAR implemented with a federated multi-task transfer learning architecture. The comparison of different aspects of this thesis with the mentioned studies can be seen in Table 2.1.

Table 2.1. Comparison with related studies.

	Papers									
Properties	[6]	[10]	[12]	[13]	[16]	[17]	[18]	[20]	[22]	Thesis
Centralized	X	X	X		X	X			X	X
Federated	X	X	X	X	X	X	X	X		X
Multi-Task	X			X				X		X
Transfer Learning	X	X			X	X	X		X	X
Personalization	X	X		X	X		X			X
HAR		X	X		X	X	X	X	X	X
Diff. Dataset/Client										X
Diff. Labels/Client										X
DeepConvLSTM									X	X

3. DATASETS

In this thesis, we utilized an ensemble of datasets (OpenHAR) [5]. This chapter will examine the details of OpenHAR along with the preprocessing steps applied.

3.1. Properties of OpenHAR

As mentioned, OpenHAR is actually an ensemble of ten different datasets, which are listed as follows

- Dataset 1: "Banos et. al." [23],
- Dataset 2: "Ortiz et. al." [11],
- Dataset 3: "Shoaib et. al." [24],
- Dataset 4: "Siirtola et. al." [25],
- Dataset 5: "Stisen et. al." [26],
- Dataset 6: "USC-HAD" [27],
- Dataset 7: "UniMib-SHAR" [28],
- Dataset 8: "HuGaDB" [29],
- Dataset 9: "RealWorldHAR" [30],
- Dataset 10: "MobiAct" [31].

In Table 3.1, the sampling rates as well as unique label amounts can be seen. Even though OpenHAR has seven different activity labels in total, there is not a single dataset that contains instances from 7 of them simultaneously.

These datasets are the state-of-the-art datasets used in different studies. Although different datasets were collected at different sampling rates, OpenHAR synchronized each dataset to have the same sampling rate at 10 Hz.

OpenHAR is actually a big set of ten smaller datasets (tabular datasets). They are all collected within different research studies and within different participants. The attributes and columns of OpenHAR subsets can be listed as (with their types in parenthesis)

- dataset_ID (integer between 1-10),
- position_ID (integer between 1-14),
- user_ID (integer between 0-66),
- activity_ID (integer between 1-99),
- timestamp (time),
- x-axis acceleration (float),
- y-axis acceleration (float),
- z-axis acceleration (float),
- magnitude of acceleration (float).

Table 3.1. Details of 10 datasets of OpenHAR.

Dataset ID	**Dataset Name**	**Unique Activity Amount**	**Unique Position Amount**	**Sampling Frequency**
1	Banos et. al	6	3	50 Hz
2	Ortiz et. al.	3	1	50 Hz
3	Shoaib et. al.	4	4	50 Hz
4	Siirtola et. al.	4	1	40 Hz
5	Stisen et. al.	3	2	50-200 Hz
6	USC-HAD	5	1	100 Hz
7	UniMib-SHAR	4	1	50 Hz
8	HuGaDB	4	2	60 Hz
9	RealWorldHAR	5	7	50 Hz
10	MobiAct	6	1	200 Hz

The OpenHAR dataset is used in different studies, such as [22] and [32]. In the former study, authors combine convolutional neural networks and long short-term memory architectures to train an activity classifier in a centralized manner. In the latter study, authors concentrate on reducing the model to a smaller "lite" version to make it usable for less-capacity devices.

We focused on human activity recognition (HAR) and position recognition, i.e., where the phone was carried, tasks. Hence, from these columns, x, y and z axis and magnitude accelerations are used as features during training, whereas activity_ID and position_ID are used as the labels. User_ID and dataset_ID are used as separators for clients in various experiments.

3.2. Preprocessing

In this section, the preprocessing of different labels is explained. Preprocessing of labels is needed because of these principal reasons:

- there were quite similar labels (e.g. sitting vs. sitting in car) which are hard or not very meaningful to classify,
- some labels had very few instances (e.g. walking at stairs),
- some labels existed only in one or two datasets (e.g. elevator up and elevator down).

3.2.1. Activities

OpenHAR includes 16 different activity labels in its original version.

In Figure 3.1, 16 different activity labels' distribution can be seen. During the preprocessing of activity labels,

- NULL labels are deleted,
- Standing, Sitting, Idling, Sitting in Car, Elevator Up and Elevator Down labels are combined into Idling,
- Walking, Walking stairs up, Walking stairs down, Walking at stairs (inc. up and down) are combined into Walking.

In Figure 3.2, the seven labels of activities of OpenHAR can be seen after the preprocessing. As we are going to separate the data into clients, we also examined the distribution of the labels for every dataset (by dataset_id).

Figure 3.4 shows the activity distributions for each of the ten datasets. The minimum activity label type count is 3, which is chosen for training as a minimum number. As we wanted to maintain the classification as a multi-label problem (in order to make the challenge remain at a certain level), a minimum of three different labels are accepted. However, each activity type does not exist in each dataset, which adds to the challenging aspect of the thesis's main problem.

3.2.2. Positions

OpenHAR consists of 14 different position labels in the unprocessed state.

In Figure 3.3, the 14 labels of positions of OpenHAR can be seen. During the data collection of some datasets, sensor position was defined without any direction (left or right), meaning that it can either be left or right, while in some cases, position was explicitly defined as left or right (denoted by L or R). In addition, in some cases, study subjects were allowed to decide the orientation of the sensor, while in some studies, orientation was fixed, which is denoted by $Free$ or $Fixed$. During the preprocessing steps

- Hip, Trouser's pocket left (fixed orientation), Trouser's pocket right (fixed orientation), Trouser's pocket left (free orientation), Trouser's pocket right (free orientation) are all combined into Pocket,
- Shin, Ankle, Foot left, Foot right are all combined into Leg/Foot.

The reason for these recombinations is either they have very few instances, or they are strongly correlated positions, from almost the same body parts.

In Figure 3.5, the seven activity labels of OpenHAR can be seen after the preprocessing. As we are going to separate the data into clients, the distribution of labels for every dataset (by dataset_id) should be examined.

Figure 3.6 shows the position distributions for each of the ten datasets. Only datasets 1, 3 and 9 have label type counts larger than or equal to 3, and as we wanted to maintain the problem non-binary (in order to make the challenge remain at a certain level), only these three datasets are used for position training. Hence, other datasets are excluded from position classification because of the lack of label types.

3.2.3. Users

OpenHAR consists of ten different datasets, each collected from a different group of participants. Hence, a general user-data percentage is not given; rather, all datasets are analyzed internally.

Figure 3.7 shows the user distributions for each of the ten datasets. As we explain in Chapter 6, different datasets will be associated with a different client for federated learning. Only in one experiment, dataset-9 is separated into its users, and those users' data are considered to be owned by different clients to train. The reasons for selecting the dataset 9 are that:

- it is the largest dataset regarding the number of instances,
- it is used for both activity and position training,
- it has a balanced instance-user distribution.

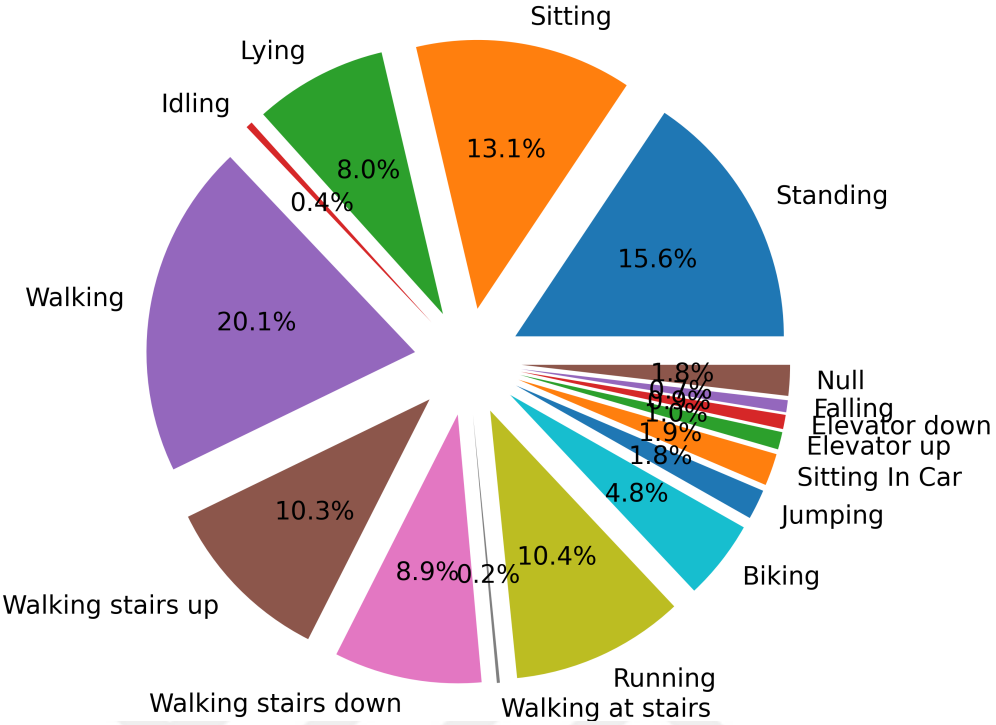


Figure 3.1. Activity label distribution of OpenHAR, unprocessed.

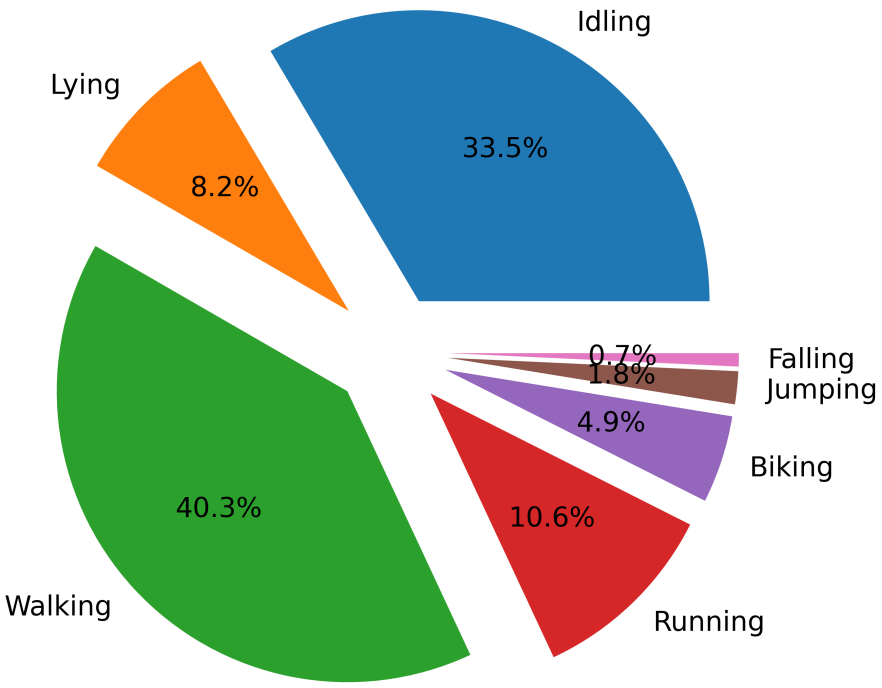


Figure 3.2. Activity label distribution of OpenHAR, after preprocessing.

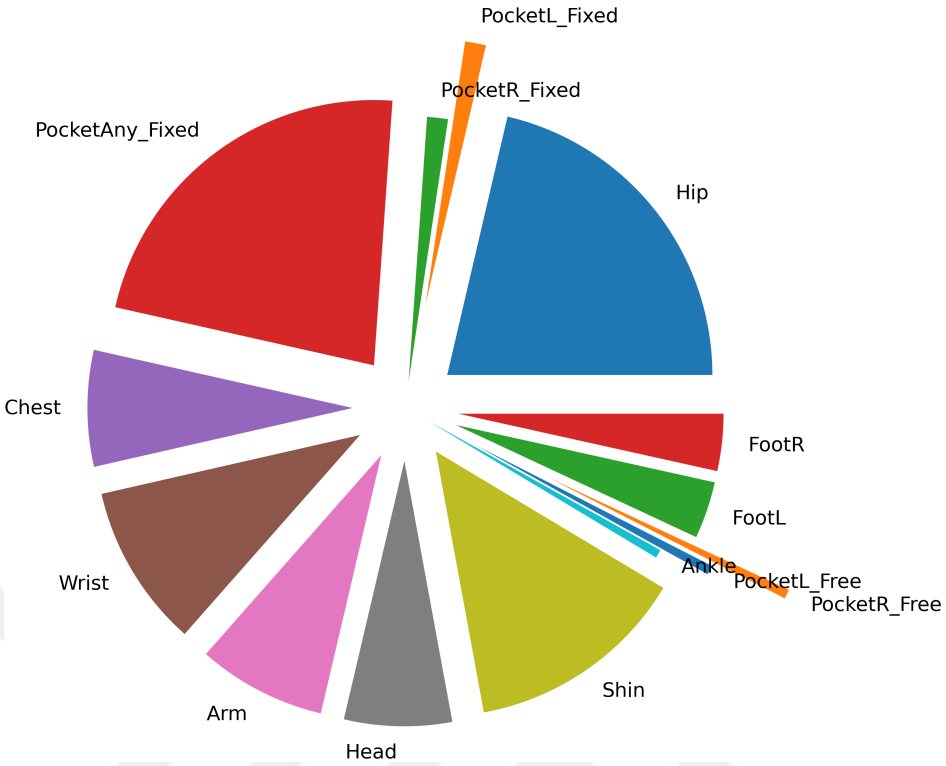


Figure 3.3. Position label distribution of OpenHAR, unprocessed.

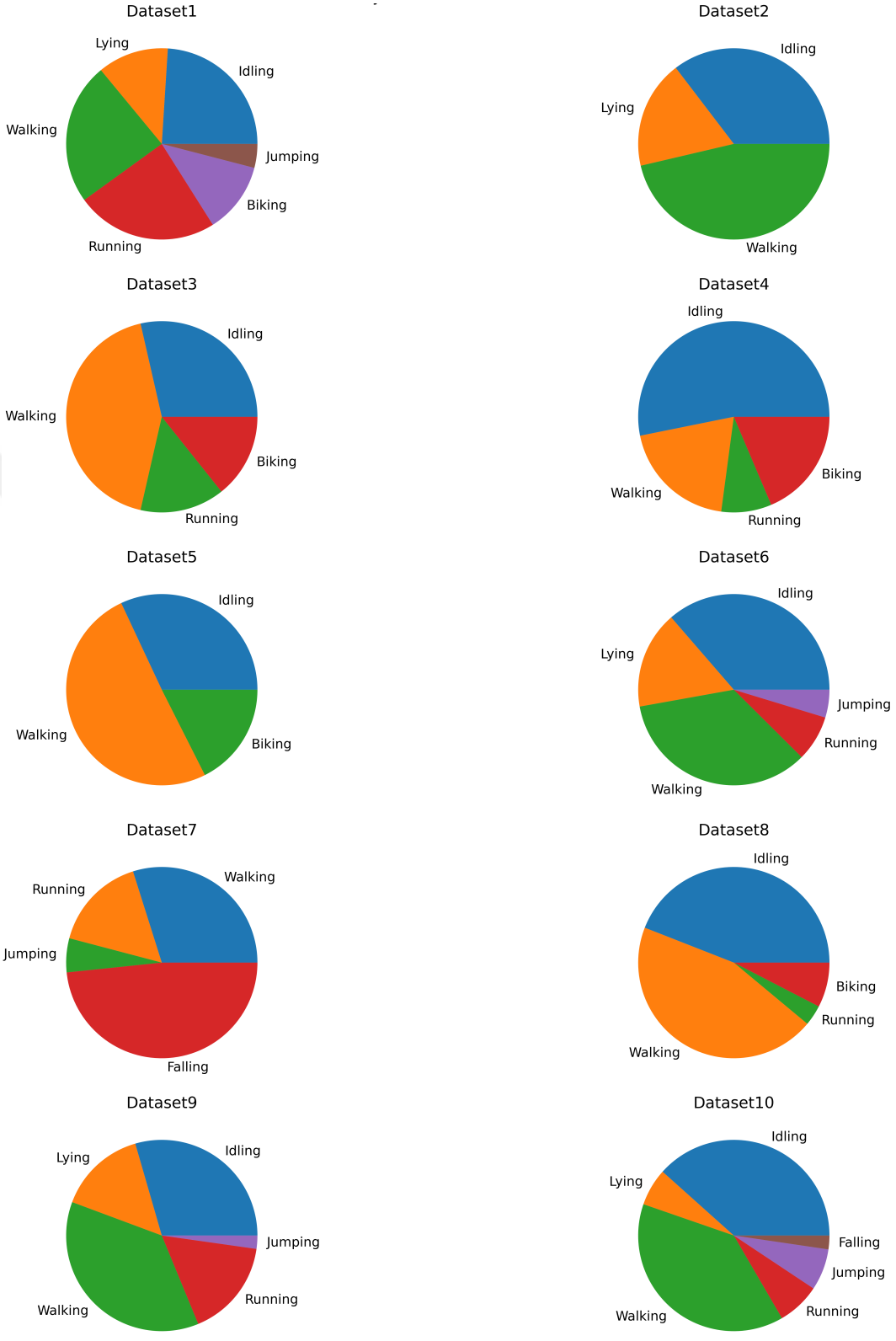


Figure 3.4. Activity label distribution of OpenHAR, after preprocessing, for each dataset.

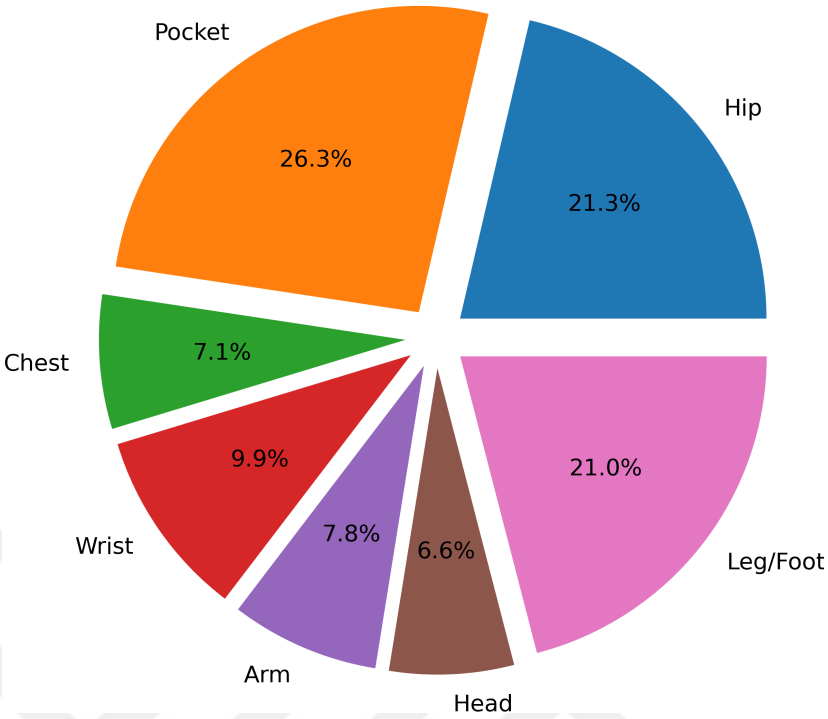


Figure 3.5. Position label distribution of OpenHAR, after preprocessing.

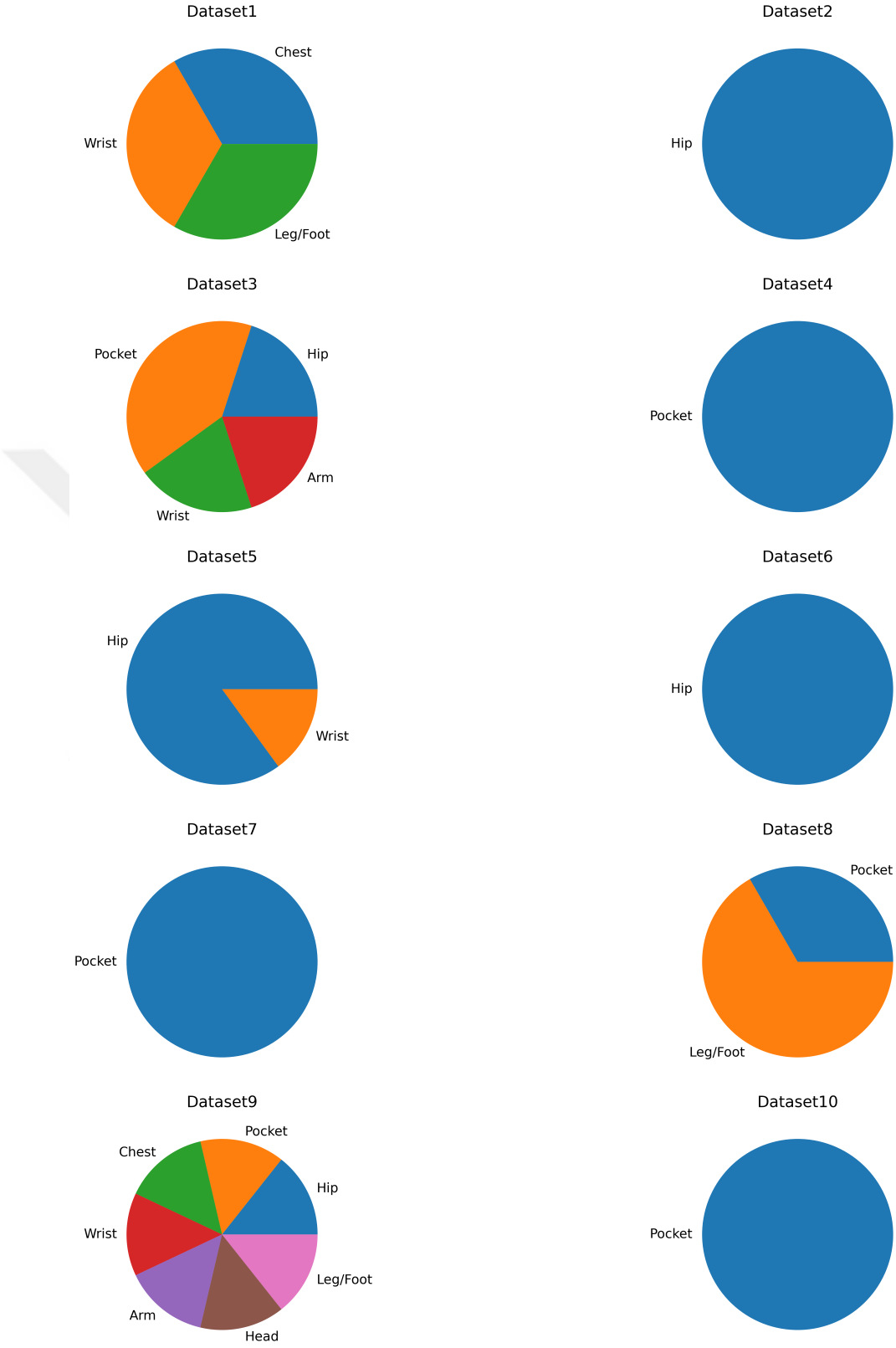


Figure 3.6. Position label distribution of OpenHAR, after preprocessing, for each dataset.

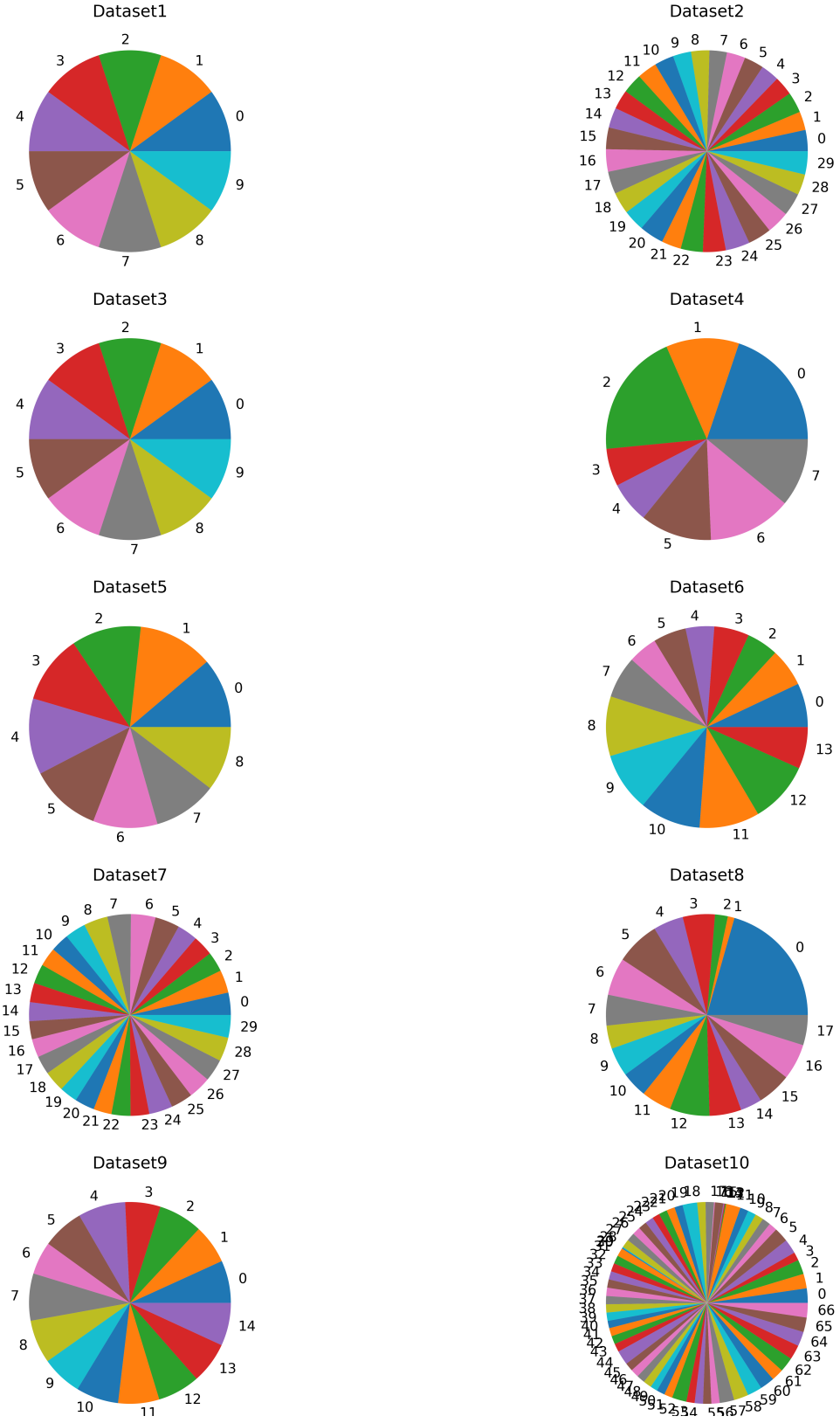


Figure 3.7. User distribution of OpenHAR, for each dataset.

4. FEDOPENHAR METHOD

In this chapter, we first present the federated learning concept along with the details of federation algorithms, then the federated simulation framework used in the thesis. The details of the FedOpenHAR framework are introduced along with the trained neural network types. Further, in this section, we highlight the details of the performance metrics used in the evaluations.

4.1. Federated Learning

Federated Learning is one of the most recent machine learning techniques, first mentioned by Google in 2016 [3]. The critical idea in federated learning is preserving privacy without any movement of private data.

Being an application of mobile edge networks, federated learning works in contrast with centralized approaches. At the time of its rise, personal data was still being shared, which was completely changed by the arrival of federated learning, which allowed for privacy. Instead of data sharing, only model parameters are shared among the participants, which helps reduce communication costs and low latency.

In this thesis, the federated learning (FL) area is chosen because of the broad current usage of mobile devices that bring an opportunity to use mobile edge networks and that FL lets the framework comply with the GDPR privacy rules.

The subject of federated learning is narrowed down to federated transfer learning, as it also contains and focuses on the personalization of the results obtained from federated learning. While many resources/participants join in the federated learning method, all participants may have different characteristics, varying from one to another. The federated transfer learning method addresses this issue efficiently.

In the transfer learning method, the knowledge is transferred from one model to another. In neural networks, this step is done by training a model within dataset A and then transferring this knowledge to introduce a different model within dataset B just by freezing the last few layers of the pre-trained model.

4.1.1. Federated Aggregation Algorithms

During the flow of open FedOpenHAR at the end of each training round, all weights (parameters) collected from the clients must be aggregated. Different techniques can perform this aggregation. The federated aggregation algorithms used in this thesis are:

- FedAVG [3],
- FedProx [33],
- FedYogi [34].

4.1.1.1. FedAVG. In FedAvg, the parameter aggregation is based on weighted averages. The central server aggregates all local model parameters into one new global model and then resends the model to a sub-group of clients. The algorithmic flow of FedAVG is given in Figure 4.1 (The K clients are indexed by k, B is the local mini-batch size, E is the local number of training epochs, η is the learning rate and C is the fraction of selected clients) [3].

4.1.1.2. FedProx. FedProx adds a regularization value to the FedAVG, and the minimization function becomes: The received signal is expressed as

$$\min_{w} h_k(w; w^t) = F_k(w) + \frac{\mu}{k}||w - w^t||^2. \tag{4.1}$$

The variables in Equation (4.1) are:

- $F_k(w)$ represents the local loss function the k-th client is trying to minimize,
- w are the local parameters we are trying to optimize the function with,

- w_t are the global weights of the server at epoch t,
- $h_k(w; w_t)$ is the objective function that client k is trying to minimize (the sum of the local loss function and the regularization term - i.e. the proximal term we have been talking about),
- μ is the regularization value.

4.1.1.3. FedYogi. FedYogi is a derived version of FedAVG combined with Yogi optimizer, which can also be used as a non-federated optimizer. The yogi optimizer has some unique parameters which allow to customize the optimization, such as:

- η: Server-side learning rate. Defaults to 10^-2.
- η_l: Client-side learning rate. Defaults to 0.0316.
- β_1: Momentum parameter. Defaults to 0.9.
- β_2: Second moment parameter. Defaults to 0.99.
- τ: Controls the algorithm's degree of adaptability. Defaults to 10^{-3}.

4.1.2. Federated Simulation Framework

Federated learning works with entities such as clients and servers. All participants need communication in many steps in order to function properly. During the experiments, these entities must be simulated, as using tens of different devices would be impractical.

In order to handle the simulation task, a simulation environment is required to facilitate the computational and communicational load. Many frameworks have been tested, such as PySyft [35], PaddleFL [36], TensorFlow Federated [37] and Flower [38]. However, the first three of them were found unsuitable because of either of these factors [39]:

- the framework does not incorporate PyTorch, which is the selected main machine learning framework of the thesis,

- the framework lacks a live community to collaborate,
- the framework has poor documentation or none.

Ultimately, the Flower federated learning environment [40] is chosen as a suitable framework. Flower provides an open-source framework that operates independently of device specifications. This framework facilitates scenarios such as client-to-client communication, model parameter distribution, and aggregation during simulations. The benefits of Flower as a framework are that it supports not only PyTorch but many other programming languages/interfaces as well, is open-source, has a live community that collaborates often, receives frequent updates, and has deep documentation.

4.2. FedOpenHAR

In this section, we introduce the FedOpenHAR architecture. It uses a Federated Multi-Task Transfer learning approach inspired by the architecture of [6], which is designed for image recognition tasks.

The main layered hierarchy can be seen in Figure 4.2. Every time a layer's training is finished, its parameters are frozen, and the training is applied to the subsequent layers.

The action flow of FedOpenHAR's progress is presented in Figure 4.3, and can be ordered as follows:

(i) First, a client is chosen to act in the role of the server.

(ii) The clients arrive and connect to the server (via a simulation or port-socket connection) (Step 1 in Figure 4.3), and each client aims to classify one task at a time (however, every client can choose their own task, hence, multi-task).

(iii) Algorithm starts with weight initialization. This initialization is called pretaining (Step 2 in Figure 4.3), and it is done on the server end at the very beginning of the whole algorithm. This step constitutes the pre-trained layers.

(iv) Second, common layers are obtained by:

- After the pretraining, all clients start with the same weights to train their models with their own data (Step 3 in Figure 4.3). They realize this training until a desired (preferably low) number of epochs (called per-client training epochs).
- After the training at the clients' end, all model parameters are sent to the server to be aggregated (Step 4 in Figure 4.3). Details of his aggregation will be presented in the next section.
- Following the aggregation (Step 5 in Figure 4.3), the updated global-model parameters are resent to the clients, and all clients restart the training at their end.
- This process (Step 3-4-5 in Figure 4.3) continues iteratively until a desired number of communication rounds is achieved.

(v) Following the common layers, in order to obtain task-specific layers, the following steps will occur separately for each task:

- All clients for a specific task start with the same weights from the common layers and train their models with their own data (Step 6 in Figure 4.3). They realize this training until a desired (preferably low) number of epochs (called per-client training epochs).
- After the training at the client's end, all model parameters are sent to the server to be aggregated (Step 7 in Figure 4.3).
- Following the aggregation (Step 5 in Figure 4.3), the updated global-model parameters are resent to the clients, and all clients restart the training at their end.
- This process (Step 5-6-7 in Figure 4.3) continues iteratively until a desired number of communication rounds is achieved.

(vi) After finishing the training of task-specific layers, the last step of FedOpenHAR is executed: Personalization. Personalization is realized at the very end and without federation on the clients' side (Step 8 in Figure 4.3). Each client trains the global model they obtained after the last aggregation at the task-specific layers from the server. After a desired number of training epochs, the personalization step is

complete.

(vii) All clients now have their own unique and personalized models.

At the end of each training, a "test" function starts to assess training success. During the training sections on the clients' sides, the default local optimizer is SGD (Stochastic Gradient Descent), and the criterion is Cross Entropy Loss.

4.3. Models for Training

There are three different learning algorithms used when training the models for both activity recognition and position identification tasks in the experiments:

- MLP,
- CNN [41],
- DeepConvLSTM [4].

4.3.1. MLP

MLP stands for "multi-layer perceptron" and is one of the basic neural network types. The MLP network used in our experiments consists of three layers, including two fully connected layers and one Rectified Linear Unit (ReLU) layer between them. It takes four different acceleration values (x, y, z, and magnitude) as input and provides one of seven different class labels as output. MLP is used as the first network to test whether the algorithms work as expected during the experiments since it is trained and tested quickly compared to deep learning architectures. High accuracies are not expected, as it has a shallow architecture and is unsuitable for multi-label and multi-task classification, proven by the incovergence during the initial experiments.

4.3.2. CNN

The CNN (Convolutional Neural Network) method used in the experiments is referred to as "Soft-Ordering 1-dimensional CNN" [41]. This architecture comprises

four convolutional layers. Since the data is in tabular form, there is no "proximity" relationship like pixels among them. Despite the absence of this proximity relationship, the CNN algorithm rearranges and orders the data, creating a relationship, and then sends it to the convolutional layers. It has the same input and output dimensions as MLP.

During soft-ordering 1-dimensional CNN, the feature dimension is first increased with an FC layer. The role of this layer includes providing enough data for convolution by increasing the dimension and making the frame meaningful by feature sorting. Next, data is directly reshaped into a given format (16 is chosen as the filter size). Like the basic CNN model, features are extracted in the next several 1D-Conv layers with a shortcut-like connection. Finally, the extracted features are used to predict targets through an FC layer after flattening.

The order of the layers of soft-ordering 1-dimensional CNN is:

- Dense,
- Reshape,
- Conv,
- Avg-Pool,
- Conv,
- Conv,
- Conv,
- Max-pool,
- Flatten,
- Dense.

4.3.3. DeepConvLSTM

DeepConvLSTM [4] combines Deep Convolutional Neural Network with LSTM (Long Short-Term Memory) Network. The architecture of the algorithm consists of four convolutional layers, then two dense layers and one softmax classifier layer at the end.

The convolutional layers of the first part act as feature extractors and provide feature maps of the tabular data. In contrast, the recurrent layers model the temporality of the obtained feature maps.

DeepConvLSTM network is chosen as it is proven to be effective in tabular data, especially in the HAR (Human Activity Recognition) area, by the authors in [4]. DeepConvLSTM is used within different HAR studies, as in [42], authors incorporate DeepConvLSTM with Attention modules to achieve human activity recognition. In another example, authors in [43] improve deep learning's success in the HAR area with DeepConvLSTM.

The order of the layers of the DeepConvLSTM network is:

- Conv,
- Conv,
- Conv,
- Conv-Pool,
- Dense,
- Dense,
- Softmax.

4.3.4. Metrics of Success

The default success metric used in all experiments is **accuracy**. Accuracy is calculated by

$$\frac{\text{Count of all correctly classified instances}}{\text{Count of all instances}}. \tag{4.2}$$

However, in experiments with multiple clients, one accuracy metric is insufficient to visualize a whole algorithm's success, so another metric, such as weighted average accuracy, is introduced. Weighted average accuracy can be explained as

$$\frac{\sum_{i=1}^{\text{number of clients}}(\text{Accuracy of } client_i \times \text{Count of instances in } client_i)}{\sum_{i=1}^{\text{number of clients}} \text{Count of instances in } client_i}. \tag{4.3}$$

Equation (4.3) is used during the calculation of the success of the whole FedOpenHAR algorithm or for the calculation of the success of a layer.

While implementing experiments with FedOpenHAR, getting an overall success metric is a difficult challenge. The steps are as follows:

- Each client has a test accuracy after every training epoch.
- If the client's training consists of multiple epochs, best of all, accuracies are taken as the accuracy metric for that client.
- Each client's accuracy is weighted averaged using the Equation (4.3), which gives a specific layer's overall average acccuracy.
- This process is realized for all layers, and the personalized layer's (last layer) weighted average accuracy gives the whole performance of FedOpenHAR.

As there are many steps of averaging for layers of FedOpenHAR, in order to keep the statistics simple and comparable, metrics like F1-score, precision, or recall are not used, as it would mean a whole change of calculation of metrics.

initialize $w_0 \leftarrow 10$

for each round $t = 1, 2, ...$ **do**

$n \leftarrow max(C.K, 1)$

$S_t \leftarrow$ random set of n clients

for each client $k \in S_t$ **in parallel do**

$w_t^{k+1} \leftarrow ClientUpdate(k, w_t)$

end for

$m_t \leftarrow \sum_{k \in S_t} n_k$

$w_t + 1 \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_t^{k+1}$

end for

Figure 4.1. Algorithm of FedAVG.

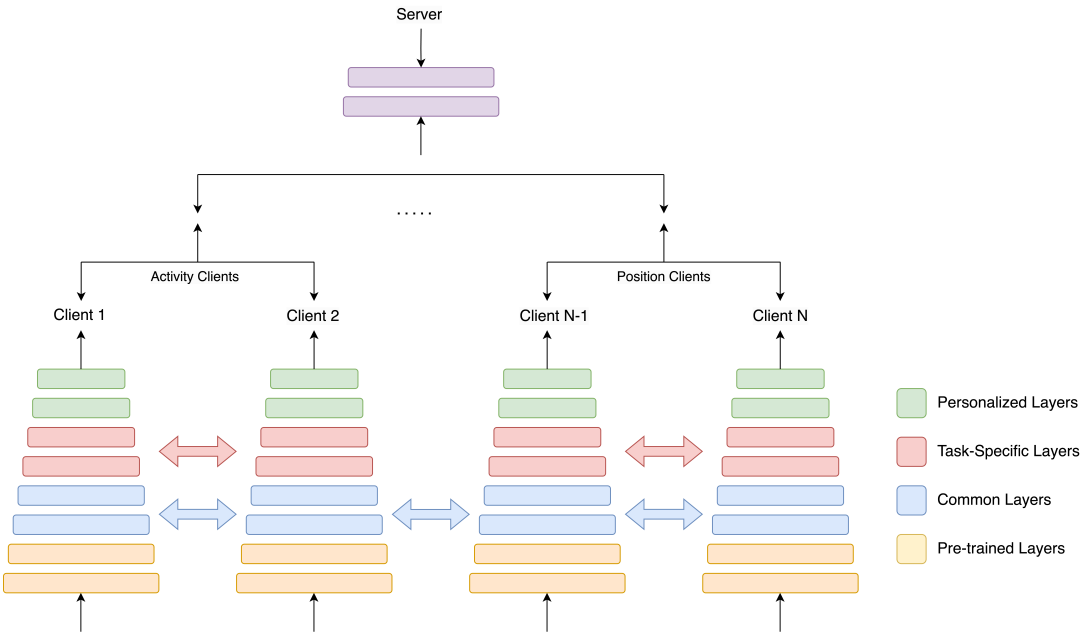


Figure 4.2. Layered hierarchy of FedOpenHAR.

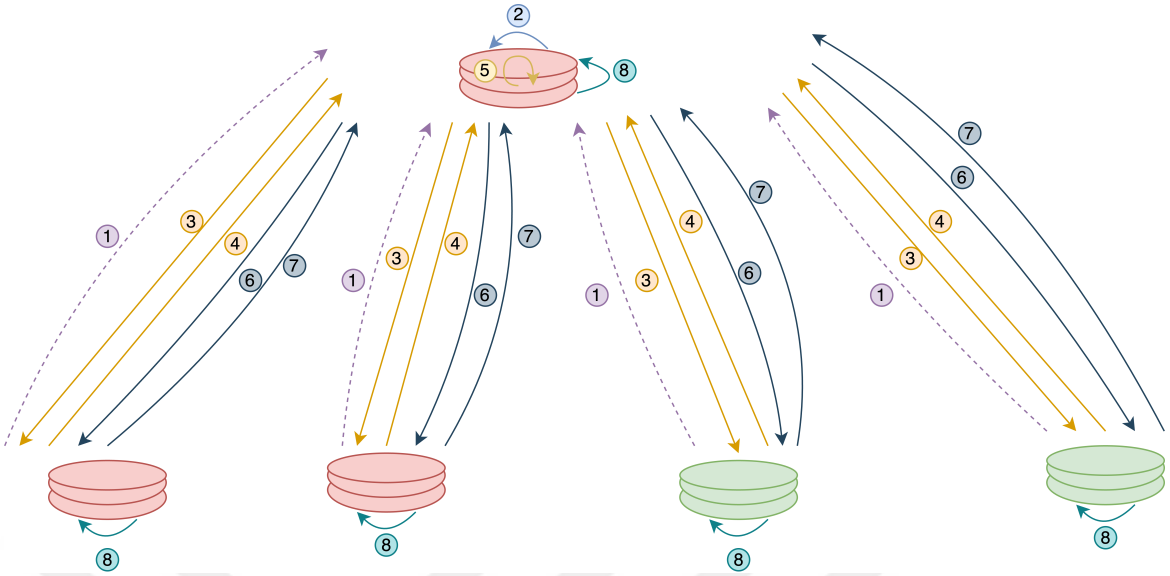


Figure 4.3. Algortihmic flow of FedOpenHAR.

5. INITIAL EXPERIMENTS

We performed different experiments to evaluate the performance of the techniques mentioned above individually and in various combinations. First, all data is combined, and training is performed in a centralized manner. Second, every client's data is trained individually, without any parameter sharing, and then simple federated learning for one task and multi-task training are explored. Finally, the results of the multi-task federated transfer learning experiments with FedOpenHAR are presented.

5.1. Centralized Experiments

No federation tool is used in centralized experiments, and for each experiment, traditional centralized structured learning steps (training and test) are realized. For all centralized experiments, data is always split into train and test with %80-%20 ratio.

During these experiments, all data from all datasets of OpenHAR are combined into one big pile, which does not conserve data privacy and contradicts the main purpose of federated learning.

Centralized experiments have been realized with three different network types: DeepConvLSTM, CNN and MLP. The training and test steps are realized separately for each label because in centralized learning, there are not multiple clients to separate the classification task. Each label type (activity and position) is tried to be classified one at a time.

In Figure 5.1, centralized experiments' test accuracies are presented for activity labels. It can be seen that the curve of MLP does not converge. Main reason of this incorvergence is that the MLP consists of only three layers (Linear-ReLU-Linear) which is not able to classify seven different activity labels. For that reason, MLP is not used for any other experiments in this study.

In Figure 5.2, centralized experiments' test accuracies are presented for position labels. Although the CNN graph seems to converge, the low accuracy puts a question mark on its success. To clarify, the inter-label accuracy is analyzed with a confusion matrix.

During the examination of the confusion matrix, it is seen that the network tends to favour one specific label, which can not be called proper "learning". Therefore, CNN is not used for any other experiments in this study.

The comparison of the accuracies in centralized experiments can be seen in Table 5.3.3. MLP result is not given for position labels because the graph of activity labels for MLP in Figure 5.1 does not converge. This inconvergence is due to the fact that MLP method's few and simple layers (Linear-ReLU-Linear) is not enough to classify a label with seven different classes. Although the CNN method performed well in activity labels, a dramatic drop can be observed in position labels, which, upon further examination of the confusion matrix, seemed to be because of the imbalanced favour of one specific label. Between these experiments, DeepConvLSTM performed the best in both label types.

5.2. Individual Experiments

For the second class of experiments, instead of combining data into one big pile, each client (each dataset) is treated separately, using only their data, conserving data privacy. However, they cannot benefit from other clients' data. The position identification task is only trained with three clients because they are the only clients with more than two types of position labels in their data. The accuracy is reported in terms of the weighted average accuracy of all clients. It is taken as the baseline for further comparison with the federated experiments. For all individual experiments, data is always split into train and test with %80-%20 ratio. Individual experiments are realized with the best-performing network, DeepConvLSTM.

Figure 5.3 presents the accuracy plot for both labels within dataset 9. In total, 13 different graphs for 13 datasets are obtained. The 13 datasets can be explained as ten activity and three position datasets. 7 datasets are not used with position labels, as they tend to have one or two position labels, which are insufficient for classification.

The comparison of the accuracies in individual experiments can be seen in Table 5.3.3. The accuracy metric in this table is the weighted accuracy of all clients. For activity labels, ten clients' accuracies are weighted by their data size, and averaged. For position labels three clients' accuracies are weighted by their data size, and averaged. These experiments are considered as a baseline to compare with the further results.

In Figure 5.4, instead of weighted accuracies, we can see the accuracy metric of every client individually. The first ten bars are for activity clients, and the last three are for position clients. From Figure 5.4, we can say that clients' success differs much from one dataset to another. The number of label types and dataset size are the factors that influence the difference above.

5.3. Federated Experiments

For the federated experiments, the setup is similar to the individual training experiments, but this time, there is model parameter sharing, hence the federated learning. Ten datasets will now be called "clients", and those clients do not share private information from their dataset. For all federated experiments, data is always split into train and test with %90-%10 ratio. In all federated simulations, clients only have one training epoch on their devices, and simulations run until ten communication rounds are completed.

5.3.1. Federated One-Task Experiments

During these federated experiments, every label type is trained one at a time, so one simulation for activity labels (with ten clients) and another for position labels

(with three clients) is implemented.

The accuracy graph for both activity and position labels for federated one-task experiments can be seen in Figure 5.5 and Figure 5.6 respectively. The accuracy metrics seen in the graphs are calculated by the weighted average (dataset sizes as weights) of all clients' accuracies of that specific label type (activity or position) for each round. The accuracy drop from activities to positions is quite noticeable, considering the minority of position clients.

5.3.2. Federated Multi-Task Experiments

During the simple federated-multi-task experiments, rather than learning one label at a time (separate simulations), both labels are learned in the same experiment (one simulation). All the activity and position clients participate in the same federated simulation, making up to 13 clients.

Figure 5.7 shows the accuracy graph for federated multi-task experiments. Both activity and position label accuracies are represented in the same figure because both labels are learned in the same experiment; the weighted average (dataset sizes as weights) of all 13 clients' accuracies for each round can be seen in the same figure.

The multi-task accuracy results are not sufficient at first glance; upon a deeper examination of their confusion matrices, the algorithm favours some examples of labels more than others, which is not acceptable for proper learning.

The comparison of the accuracies in federated one-task experiments and multi-task experiments can be seen in Table 5.3. For one-task, accuracy of the position label is still lower than the activity label, and the overall is yet to approach centralized or individual results. For multi-task, accuracies for both labels are exactly the same, as they are found in one simulation, and weighted averaged for all existing clients, independent from labels.

5.3.3. Federated Multi-Task Experiments with Transfer Learning (FedOpenHAR)

In this section, we introduce the results of multi-task transfer learning, combining federated learning, transfer learning and multi-task learning, as presented in Section4.2. In this case, dataset-9 in OpenHAR [30] for position labels revealed the best result. Then, the next layer is trained with all 13 clients, which creates the common layers. Afterwards, two training sessions were performed for activity (10 clients) and position (3 clients) tasks, resulting in the task-specific layers. Finally, every client is fine-tuned (separately), and the last layers of their models are trained with their own data. Before the training of each layer, the previous layers are frozen.

In Figure 5.8, a layerwise comparison of weighted clients' success for all layers is presented, which is also compared with the individual setting. The personalization at the end achieved accuracy similar to the baseline centralized separately (which is the individual setting in this graph explained in Section 5.2). Even though the result looks similar to the baseline, the benefit of this layered approach is that multiple layers are obtained, allowing quicker and more successful options for possible new label types, new labels in one type, or even whole new datasets.

In Figure 5.9, a client-wise comparison can be seen to compare the individual experiments (explained in Section 5.2 with FedOpenHAR for all clients. The horizontal axis represents the clients, the first letter represents the task type (A: activity or P: position), and the number attached to it represents the dataset_id. The vertical axis represents the accuracies in percentage. Results seem quite comparable, but the scarcity of label types and the smaller size of the client's dataset are factors that negatively affect FedOpenHAR's success.

In Figure 5.10, the client-wise and labelwise comparison can be seen fused in the same graph to obtain a further deep analysis. The success of FedOpenHAR is best observable in the largest dataset (P9), and its weaknesses are generally in scarce labels

and smaller datasets. P1 and A7 seem to obtain a lower accuracy with FedOpenHAR, but not with the individual (baseline) setting. The key point is that both datasets are smaller than others by instance size, and their label type counts are scarcer than others, which is an important challenge of these datasets (can be seen in 3.1. The individual setting does not get effected as much as FedOpenHAR, because FedOpenHAR's realization of averaging and weighting multiple times in multiple epochs, disfavors the success of smaller datasets.

The comparison of FedOpenHAR's each layer's success with different experiments' results can be seen in Figure 5.11. The results of various techniques and comparisons with various layers can be seen in the same figure. FedOpenHAR performs better than the centralized and simple (unlayered) multi-task approach and achieves almost the same performance as the individual setting. The benefit of FedOpenHAR appears when new clients arrive. FedOpenHAR classifies activities and positions by default; however, if a new client arrives with a totally new label type, it can start training from the common layer. Also, if a new client wants to classify a new type of activity or position, it can start training directly from task-specific layers. These benefits are not available in the individual setting.

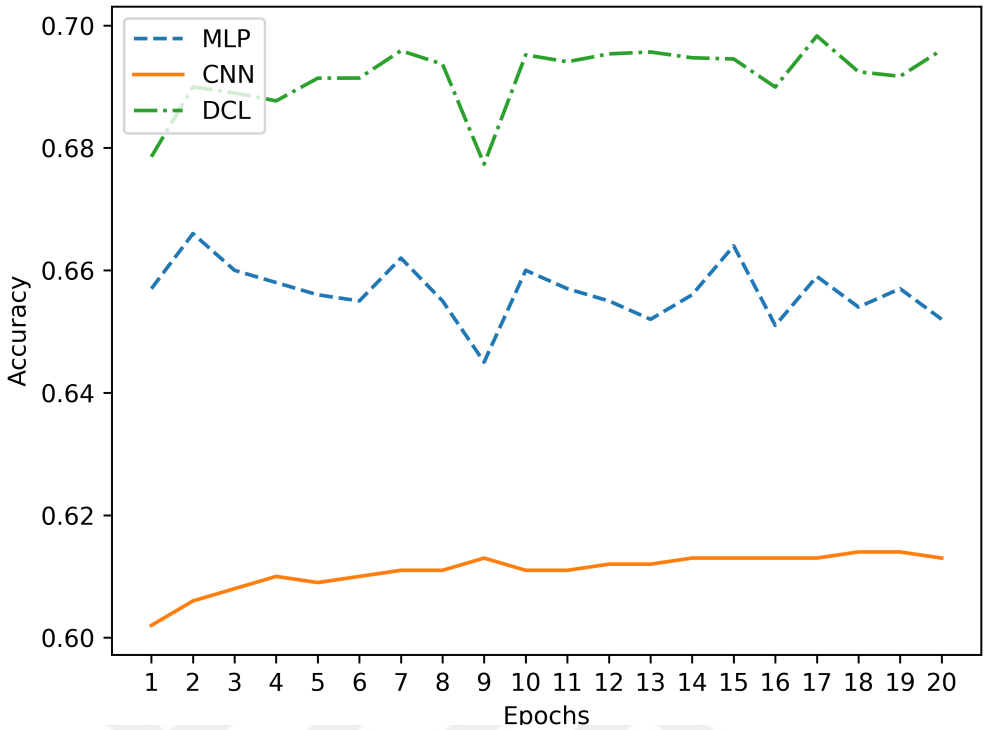


Figure 5.1. Centralized accuracies with MLP, CNN and DeepConvLSTM methods - activity labels.

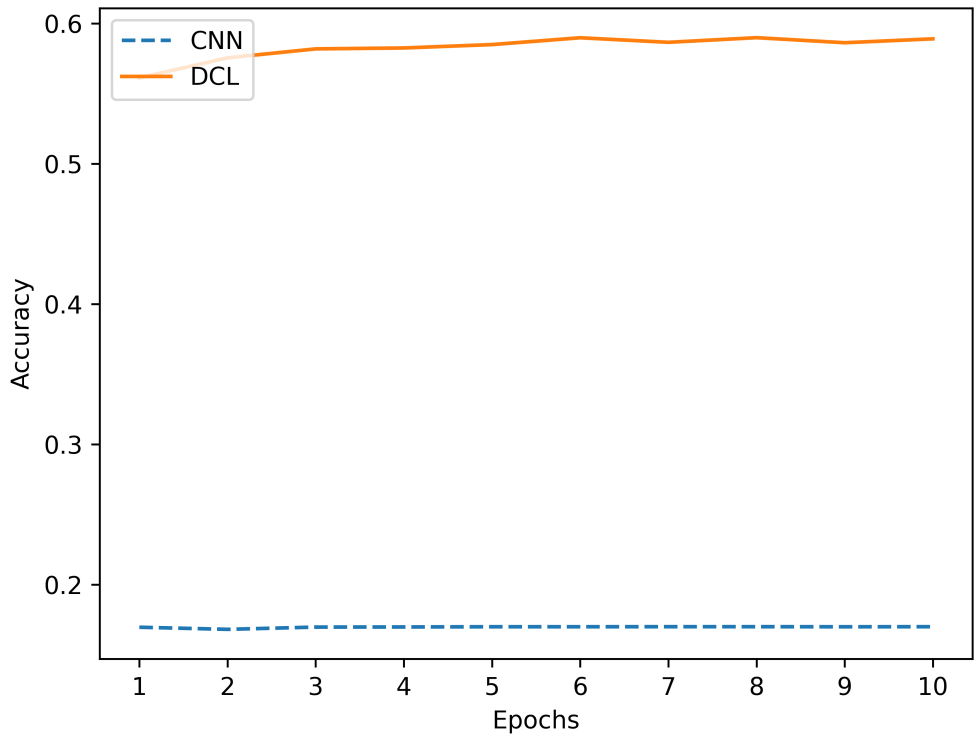


Figure 5.2. Centralized accuracies with CNN and DeepConvLSTM methods - position labels.

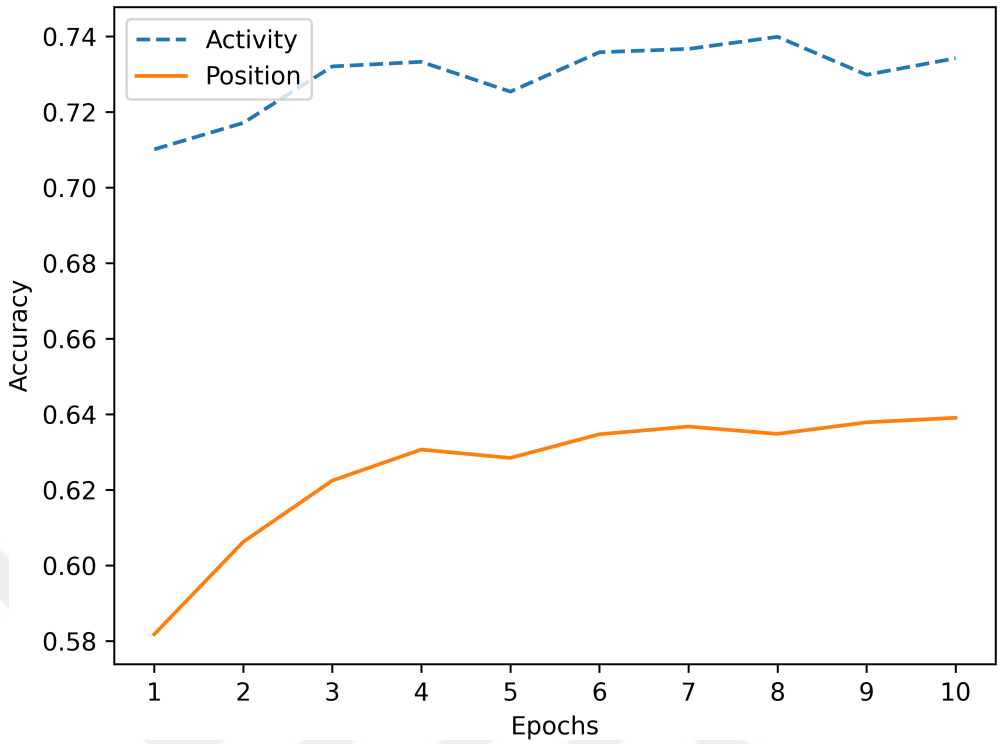


Figure 5.3. Individual accuracies with DeepConvLSTM method, for both activity and position labels in the 9. dataset.

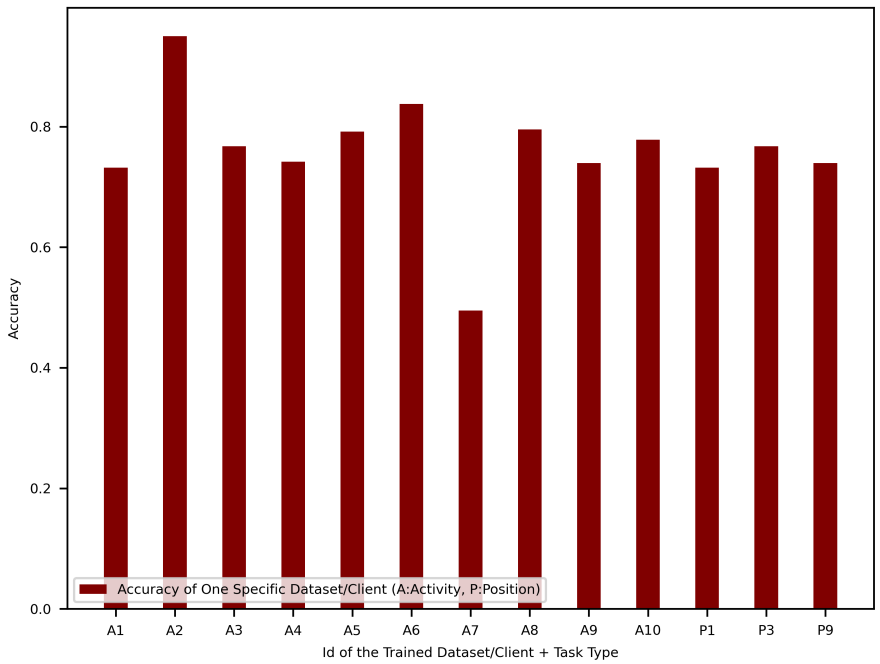


Figure 5.4. Client-wise comparison of accuracies of individual clients.

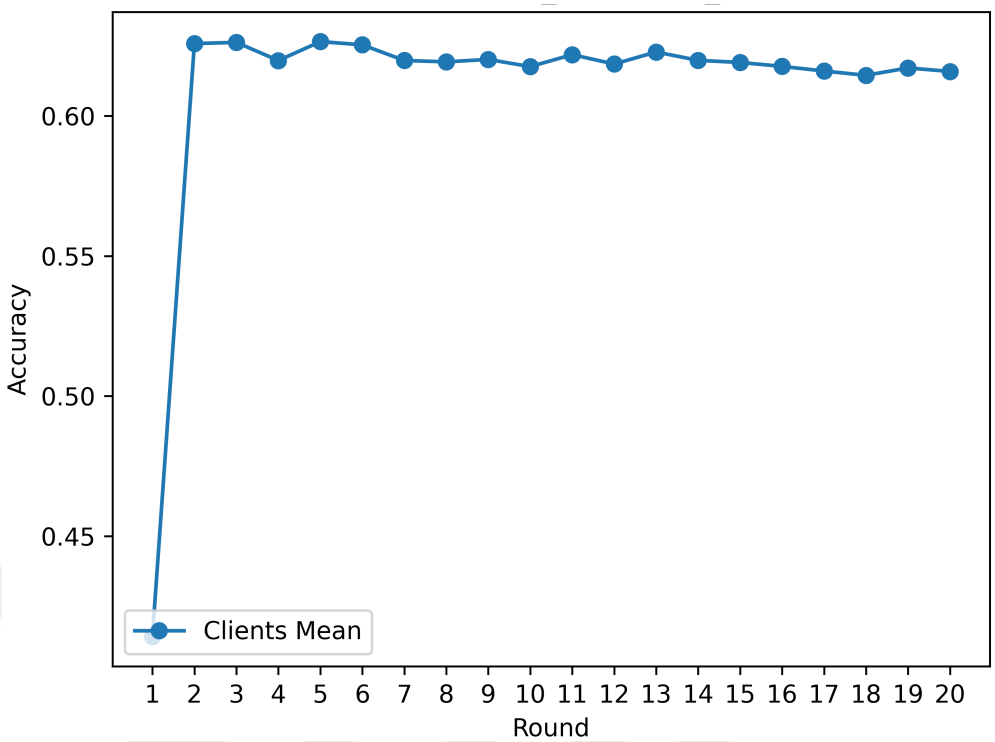


Figure 5.5. Federated one-task accuracies with the DeepConvLSTM method for activity labels.

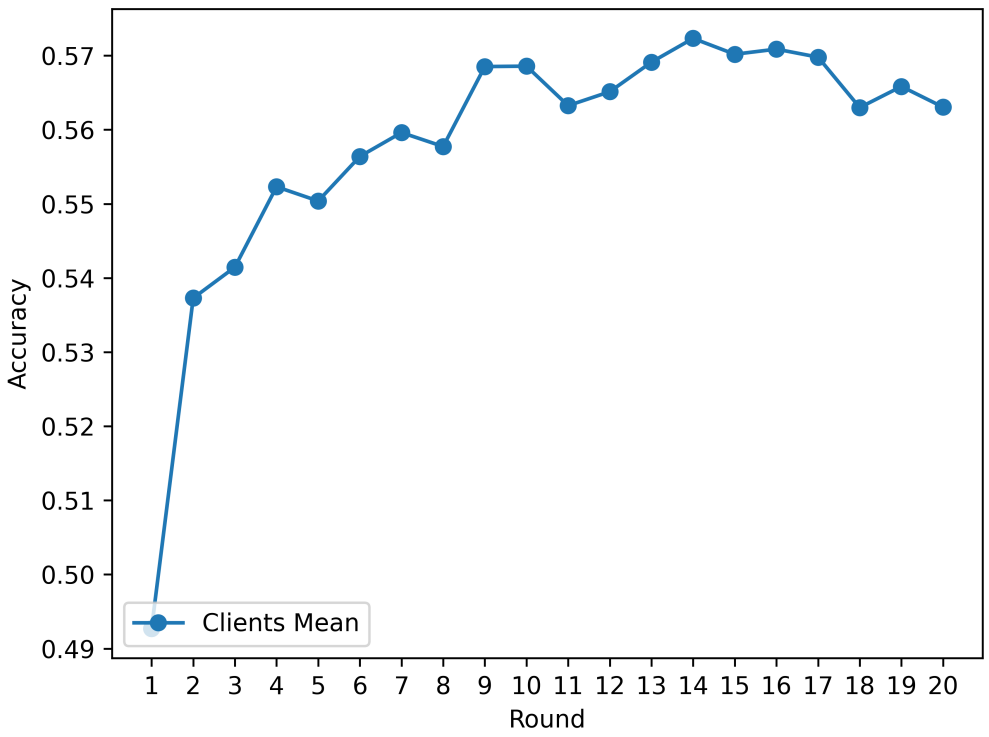


Figure 5.6. Federated one-task accuracies with DeepConvLSTM method, for position labels.

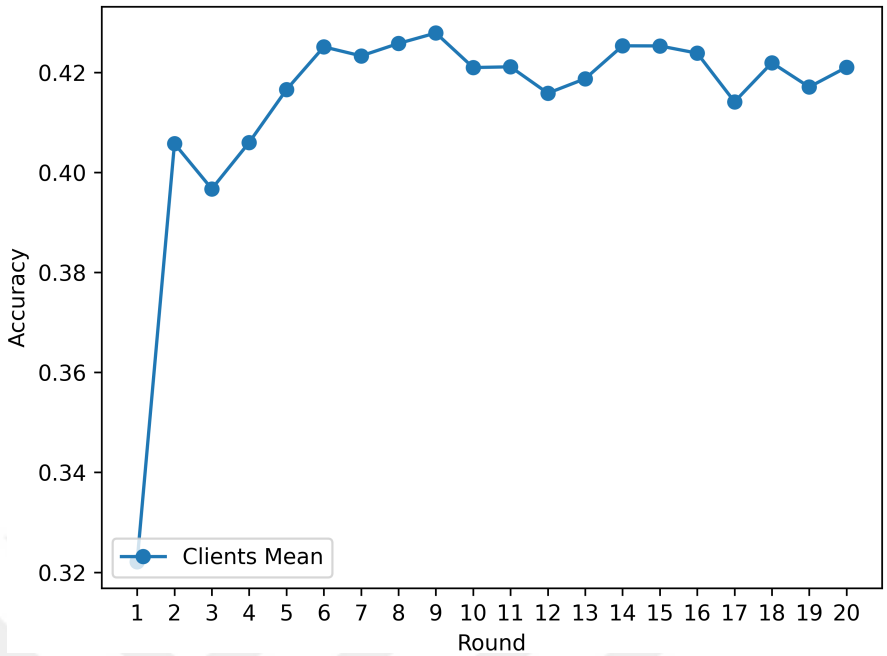


Figure 5.7. Federated multi-task accuracies with DeepConvLSTM method, for activity & position labels.

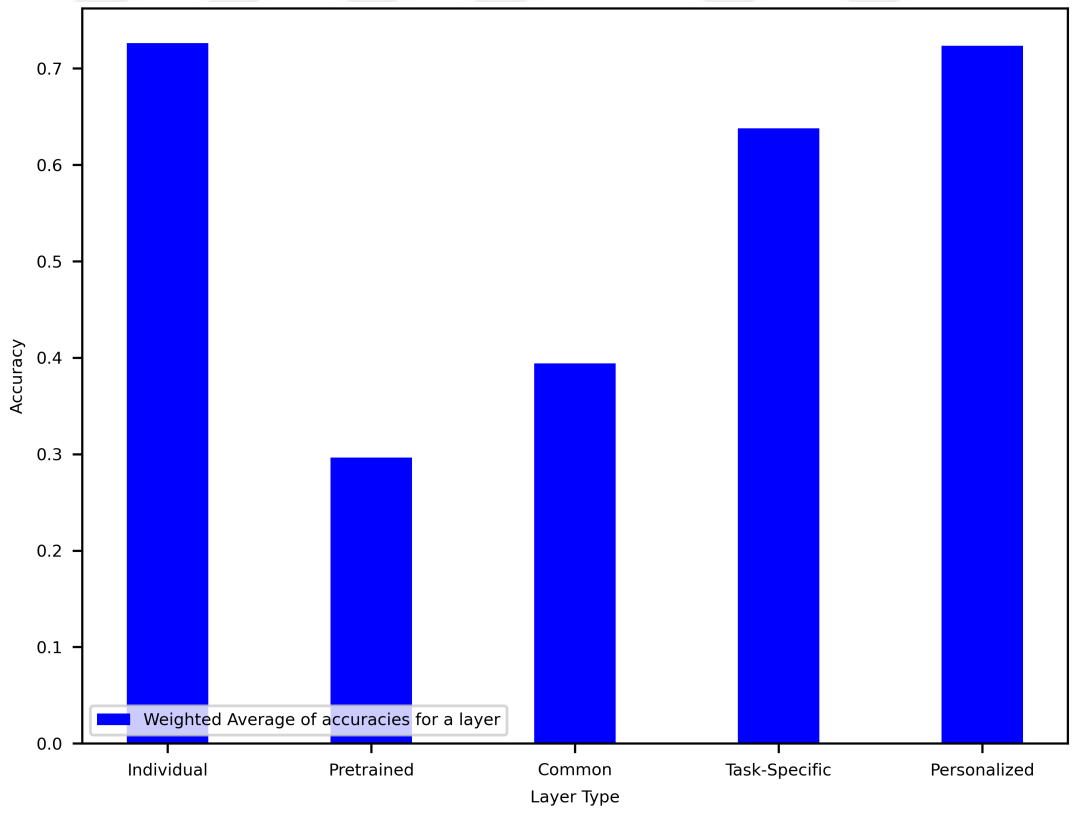


Figure 5.8. Weighted average accuracies of clients by layers and comparison with the individual experiments' average.

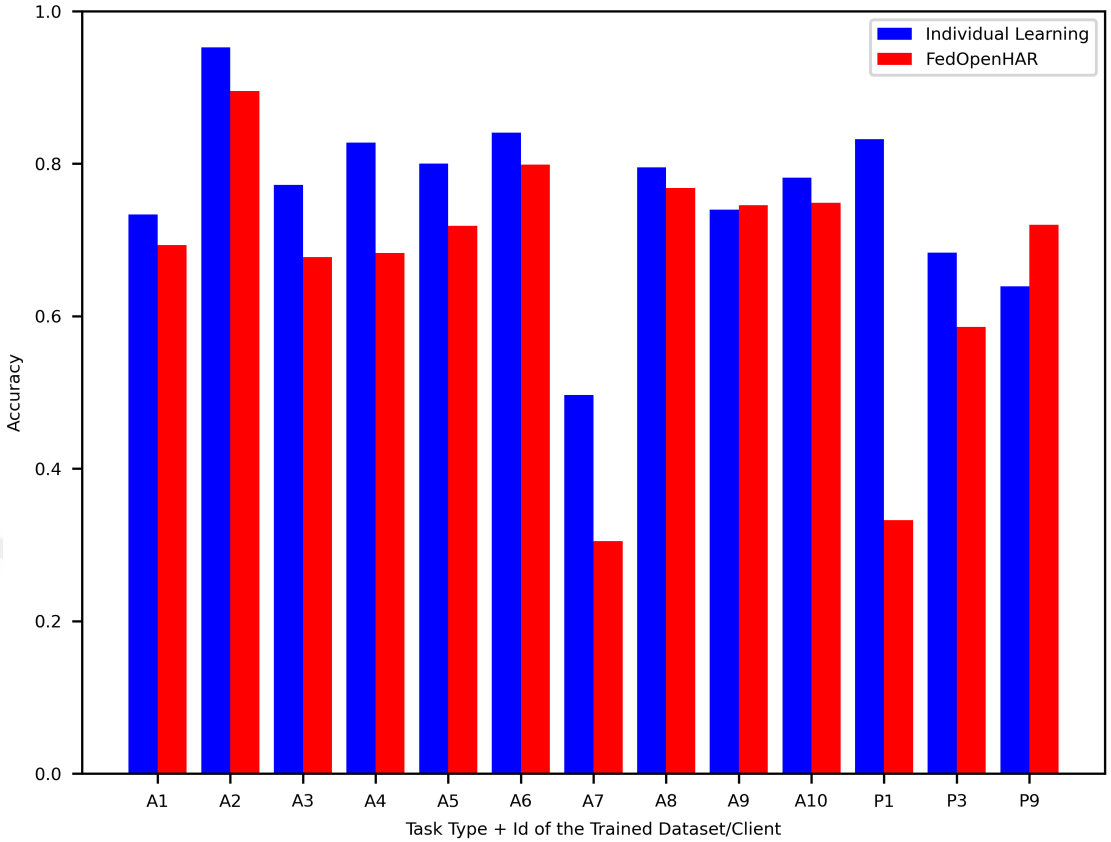


Figure 5.9. Client-wise comparison of accuracies for individual learning and FedOpenHAR.

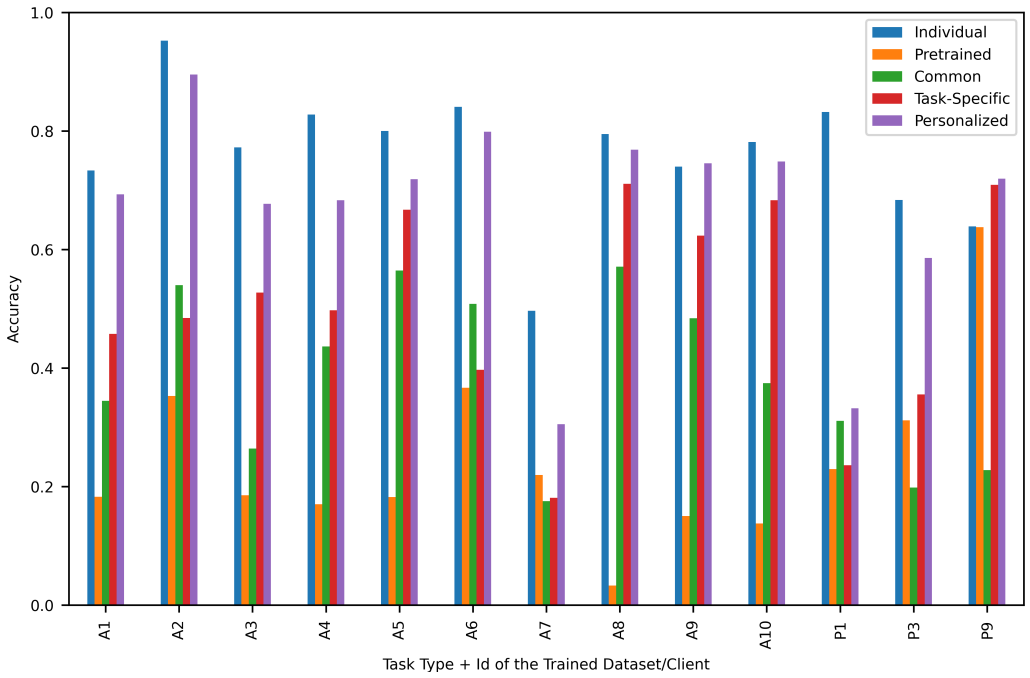


Figure 5.10. Client-wise & layerwise comparison of FedOpenHAR.

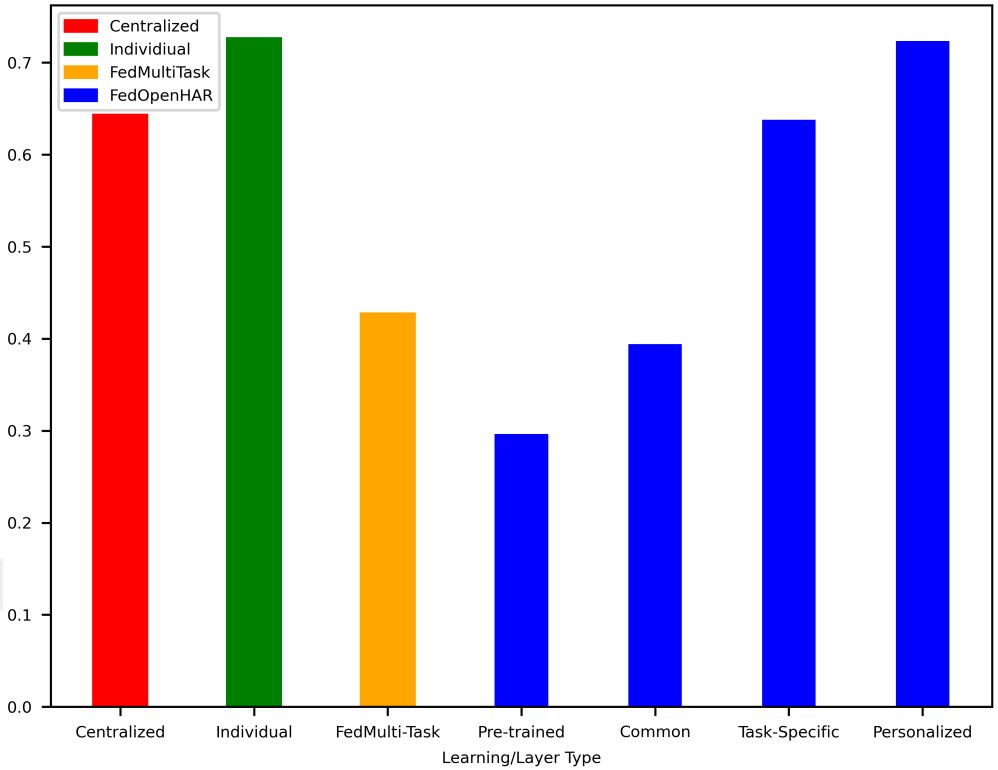


Figure 5.11. Weighted average accuracies of clients by layers and comparison with the individual experiments' average.

Table 5.1. Comparison of different centralized approaches' accuracies for activity and position labels.

	Label Type	
Methods	Activity	Position
DeepConvLSTM	**69.8%**	**59.2%**
CNN	61.4%	17.0%
MLP	66.6%	-

Table 5.2. Individual experiments' accuracy results for both labels with the DeepConvLSTM method.

	Label Type	
Methods	Activity	Position
DeepConvLSTM	76.7%	65.0%

Table 5.3. Weighted accuracies for federated experiments.

	Label Type	
Methods	Activity	Position
Federated One-Task	76.7%	65.0%
Federated Multi-Task	42.7%	42.7%

6. FURTHER EVALUATION OF FEDOPENHAR

Given that the FedOpenHAR approach performs best among the other techniques, we analyze the impact of different parameters, features, optimization techniques and averaging methods in this section.

6.1. Impact of the Number of Pretraining Epochs

The pretraining of the model is done at the local server in a centralized manner. This pretraining can be run at multiple epochs. This experiment explores the impact of the number of epochs from 1 to 10 on the classification performance. Pretraining is a step done in a centralized manner on the server (which is actually a client), and as in federated learning, we generally work on devices with smaller computational capacities; more numbers for pretraining epochs are not explored, which would require more capacity, more time and more resources. Another point is that as the number of pretraining epochs increases, the simulation tends to move from federated learning to centralized learning, which is not the main purpose of this thesis. The dataset to pretrain is chosen as dataset 9, as it has a balanced distribution of activity and position labels. Also, it is the biggest dataset by the instance amount, which makes dataset 9 a representative dataset for OpenHAR.

As seen in Table 6.8, even though increasing the pretraining epochs improves the success of pretrained layers, the personalized layers benefit less or do not benefit at all from pretaining. As we work with multiple clients who differ from one to another, more pretraining does not directly mean more success, which can be observed by the slight drop in the last layer's accuracy.

6.2. Different Number of Personalization Epochs

The personalization of the model is done on each client's side at the very end of the whole algorithm. In contrast to the pretraining, which is done at the very beginning of the simulation and aims to initialize the weights of the global model, personalization is done on the client side, separately for each client, with their own data. The training can be run at multiple epochs. This implementation explores numbers of personalization training from 1 to 10 epochs. For ten epochs, the results are given for the "best of 10 epochs".

As seen in Table 6.8, the increase in personalization epochs offers only a slight accuracy gain. From Table 6.8, we can conclude that increasing the personalization training epochs does not impact overall performance; even one epoch is sufficient to achieve a noticeable success. However, if the personalization epochs drop to 0 (meaning no personalization), the success of the whole simulation (success of task-specific layers) drops drastically.

6.3. Optimizing All Layers vs. Only the Last Layer After Transfer Learning

During the personalization step, the optimizer can be set to optimize (not train) only the last layer of the trained network or all layers. In this context, layers do not mean the layers of federated architecture but rather the layers of the neural network that is being trained. In this implementation, both optimization approaches are explored.

As seen in Table 6.8, optimizing all layers vs optimizing only the last layer during the personalization does not affect overall success; the increase can be considered minimal. This ineffectiveness is because this optimization only affects the last part of the whole simulation, which can be considered a relatively small part of the whole simulation.

6.4. Changing the Federated Averaging Algorithm

As mentioned, this thesis's default federated aggregation algorithm is FedAVG. However, there are also different algorithms for federation. We changed the federation algorithm to FedProx and FedYogi for comparisons. FedProx adds a regularization parameter (μ) to the FedAVG, and various regularization values are explored in this section. For FedYogi, the federated averaging algorithm uses the federated version of the centralized Yogi optimizer. The properties of these methods are explained in Section 4.1.1.3.

As seen in Table 6.8, regularization with FedProx with $\mu = 0$ equals the FedAVG method, which is weighted averaging without any regularization parameter, and the other regularization values do not have a major effect on the overall performance. FedYogi performs the worst among all other federated averaging algorithms, and FedAVG still fits best in all of the federated averaging algorithms. FedAVG is one of the most common averaging algorithms used in the literature and consists of basic weighted averaging for aggregation. However, FedProx adds one regularization parameter, and FedYogi adds four different parameters to the aggregation. The variation of the algorithm and those parameters did not improve the results, but a more simplistic FedAVG achieved similar or better results. This is due to the fact that whole FedOpenHAR algorithm consists of various little steps and each of them containing multiple averaging of results. This much averaging were not compatible with the more developed aggregation methods.

6.5. Changing the Layer Sizes in the Federated Hierarchy

Our proposed algorithm FedOpenHAR consists of 4 different layers, which are pretrained, common, task-specific and personalized. In the meantime, FedOpenHAR uses DeepConvLSTM as the major training network, consisting of 14 layers. It is an important question of how many layers of FedOpenHAR architecture will correspond to the layers of the DeepConvLSTM network. During this implementation, 4 and 8-

layer sizes are explored for each layer type of FedOpenHAR. Default values for each layer are:

- Pretrained Layer size = 4,
- Common Layer size = 4,
- Task-Specific Layer size = 4,
- Personalized Layer size = 2.

When a specific layer size is set to 8, all others are set to 2. As seen in Table 6.8, changing the layer size to 8 rather than the default size of 4 does not increase the success of FedOpenHAR. The default balanced distribution of the layers worked better than the other distributions. By thinking from the reverse, the over-contraction of layers resulted in poorer performance.

6.6. Using Various Training Percentages

For structured learning, data in each client was split into train and test parts. The default value was %90. This set of experiments explores the impact of different percentages on recognition performance.

Figure 6.1 presents layer-wise accuracies for different training percentages. The drop in accuracies starts at 20% and at 1%, FedOpenHAR performs even better than the individual training. However, percentages like 1% and 0.1% are considerably small portions, and as they are selected randomly, their performance results may vary each time. Hence, we could not say that the results obtained are completely reliable or underfitting. However, between 20% and 80% of training data, FedOpenHAR does not lose its effectiveness. The test set itself, or its size, does not change in each experiment. At the end of these implementations, we can proudly say that FedOpenHAR performs similarly to the maximum training percentage even with less data available, which is an important benefit as labelled data for training can be scarce and difficult to obtain in real-life experiments. The 0.1% column is given as an extremum example, in order

to see at which extent the accuracy drops drastically.

6.7. Using FedOpenHAR in a Single Dataset

The FedOpenHAR algorithm works well with clients holding different datasets, which is a challenging case. Its performance can also be tested with a single dataset split into different clients. The split can be performed in different ways, such as:

- random division of dataset to the clients: the split of data to the clients can be completely random,
- ordered division of dataset to the clients: the split of data to the clients can follow the order in the dataset,
- division of data by user-id: the split of data to the clients can be done by the user-id.

In Figure 6.2, a layer-wise and client-wise analysis of accuracies for the random division of 20 clients for dataset 9 [30] is given. During previous experiments, ten different datasets existed, each with two types of labels: activity and position. This combination made up to 20 clients in total. Seven position clients were not used as they did not have sufficient types of position labels. There is no such case during the single dataset approach, so the single dataset is divided into 20 clients. The ten left clients, starting with the letter A, represent the activity clients (clients who try to recognize activity labels). In contrast, the ten right clients starting with the letter P represent the position clients (clients who try to recognize position labels). The results are better than the experiments on all datasets presented before. This is not quite surprising as there are no different data sources but only one, and this is a relatively balanced dataset. Figure 6.2 shows that the accuracy of position labels for predicted layers is higher on average than the activity labels, which can be explained by the balanced distribution of position labels in dataset 9, whereas the activity labels are unbalanced as shown in Figure 3.4 before. The last layer's accuracies (personalized layers) stay at a similar level for each client because personalization is realized after

the whole federation and aggregation steps finish, and after that, there is no federation and averaging in the personalization steps, as it is done individually at each client. The federation helps all clients to achieve a certain level, and personalization adds a unique boost to each client.

The distributions of activities and positions of dataset 9 can be seen in Figure 6.3 and Figure 6.4. Position label distribution is quite balanced between users but activity label distribution is less-balanced.

We also explore the effect of different client split approaches besides the random. As seen in Table 6.6, ordered division, which divides the data to the clients in the order that appears in the dataset, performs the worst in dataset 9. Random division results are quite similar to the multi-dataset approach and division by user-id even outperforms the previous results of FedOpenHAR on different datasets. This is quite intuitive as users can be considered as a natural separation of clients.

6.8. Different Ratio of Clients Sampled at Training

Not all clients may be available at the training rounds or late to send their model weights. In this section, we experiment with randomly selecting the number of participating clients with different percentages at each federation round. These experiments are carried out in the whole OpenHAR dataset.

In Figure 6.5, weighted average accuracies of different ratios for client sampling are given, and as the selection drops to 80% the algorithm even performs better. This can be explained by the randomness of the sampling. As the percentages drop even further, the success drops. During this scenario, while a percentage of clients could participate in the training rounds, all clients participated in the testing part. Test results were lower as some tested participants were not present in all training epochs. In order to minimize the effects of clients dropping, as stated in [7], slow-responding clients (high latency, slow connection) are not to be selected during the aggregation.

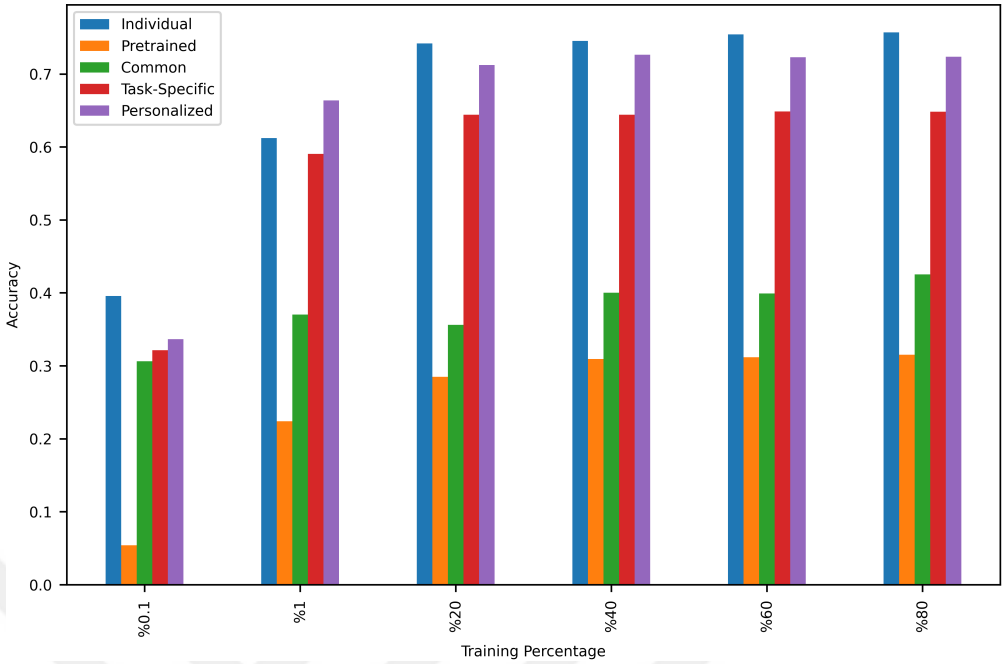


Figure 6.1. Weighted average accuracies of clients by layers separated with different training percentages.

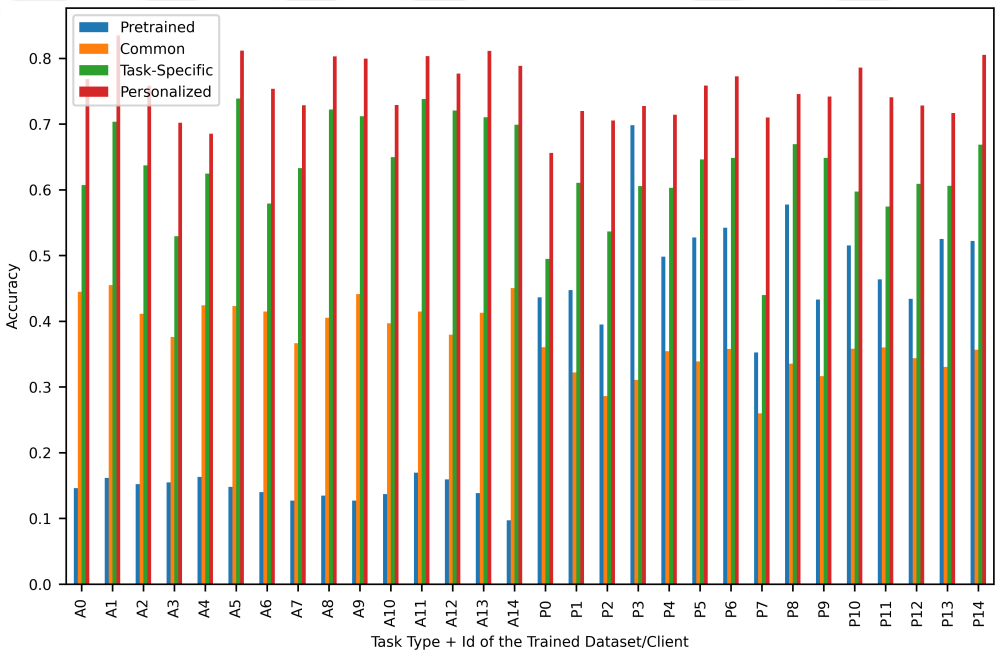


Figure 6.2. Weighted average accuracies of clients by layers separated with user-wise division in dataset 9.

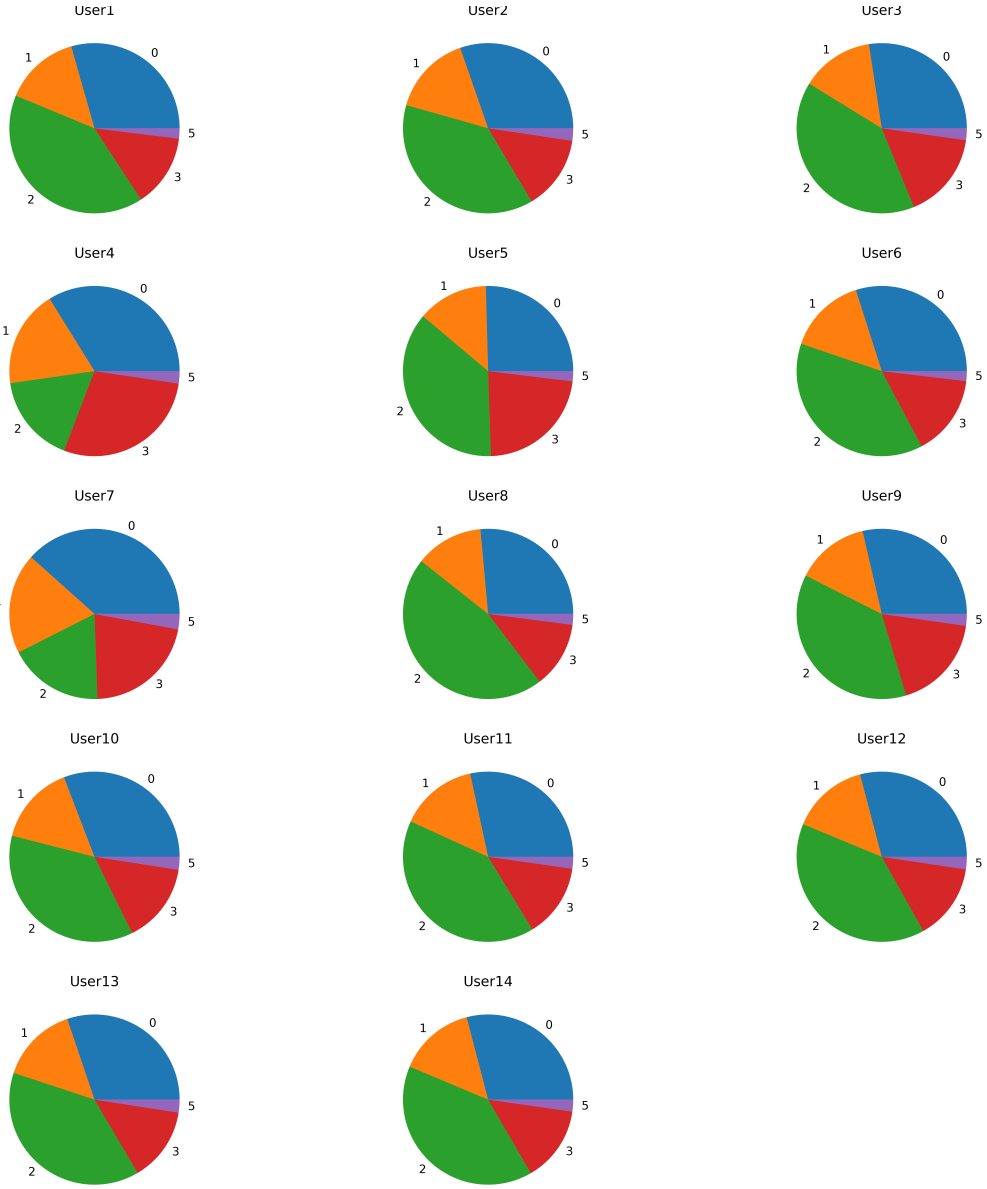


Figure 6.3. Activity distribution for userwise division of dataset 9.

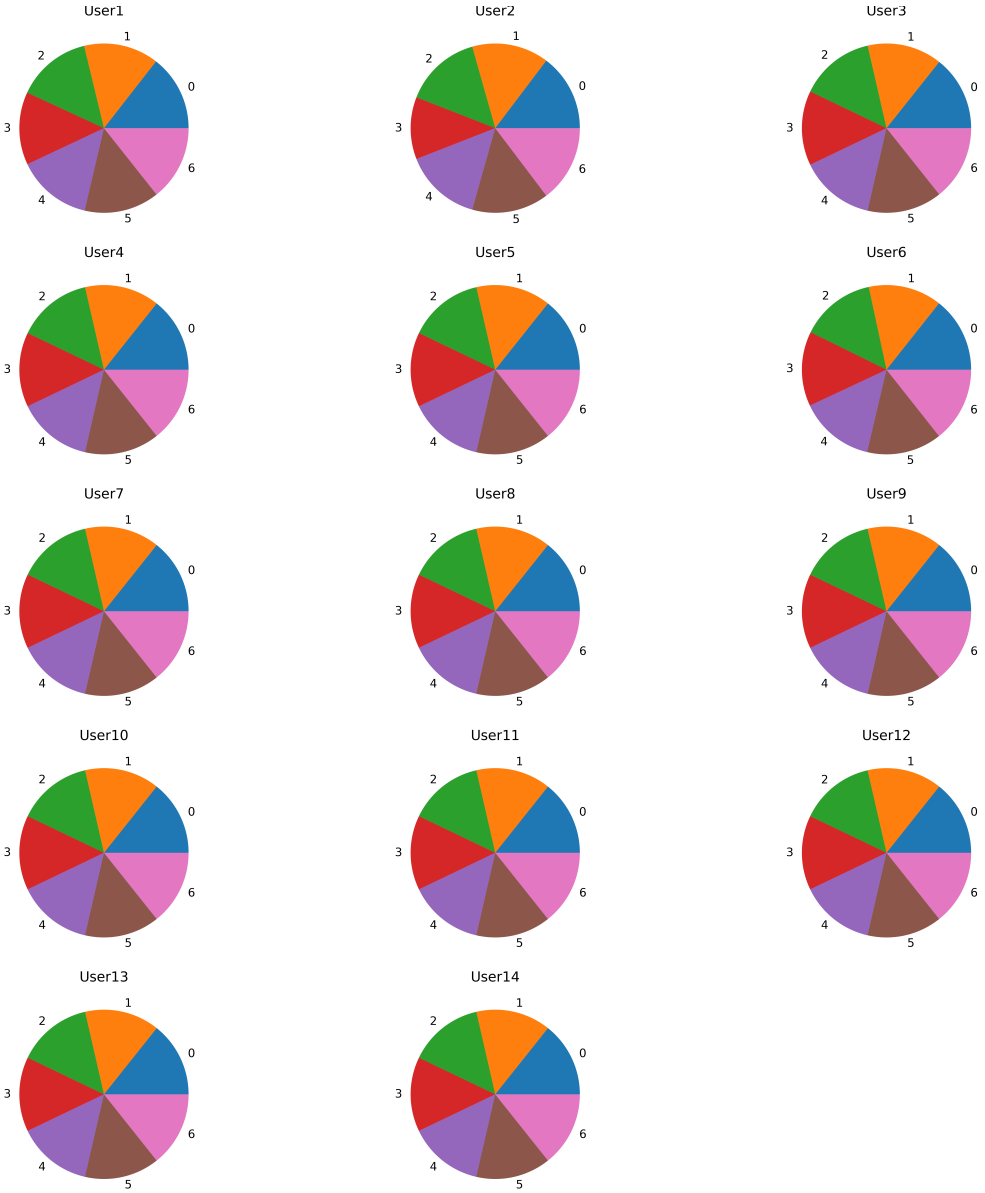


Figure 6.4. Position distribution for userwise division of dataset 9.

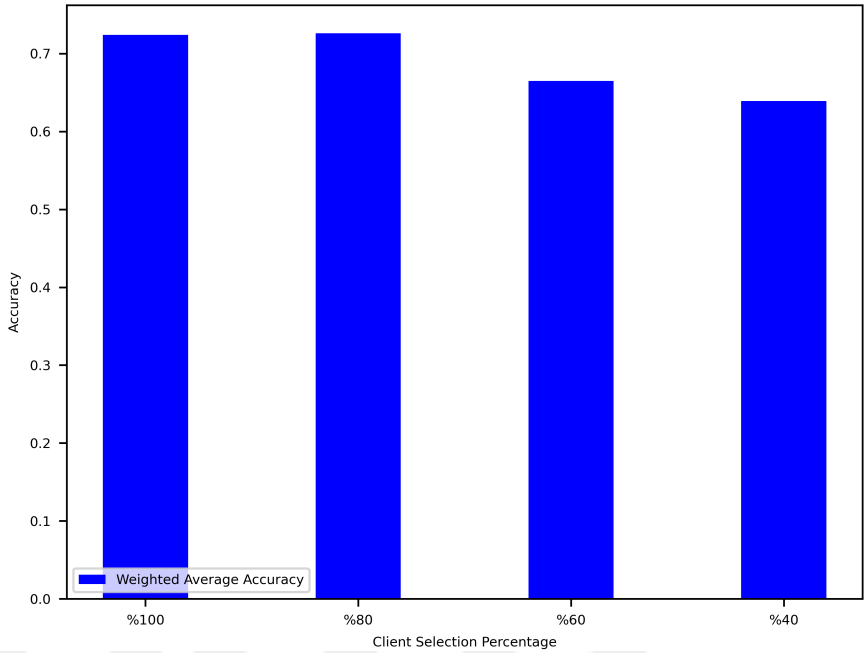


Figure 6.5. Weighted average accuracies of clients with different client selection percentages.

Table 6.1. Weighted accuracies for different pretraining epochs.

	Number of Pretraining Epochs	
Layers/Baseline	1 Epoch	10 Epochs
Individual (Baseline)	72.6%	72.6%
Pretrained	29.6%	31.1%
Common	39.4%	38.1%
Task-Specific	63.8%	63.8%
Personalized	**72.4%**	71.8%

Table 6.2. Weighted accuracies for different personalization epochs.

	Number of Personalization Epochs	
Layers/Baseline	1 Epoch	10 Epochs
Individual (Baseline)	72.6%	72.6%
Pretrained	29.6%	29.6%
Common	39.7%	39.4%
Task-Specific	64.6%	63.8%
Personalized	72.2%	**72.4%**

Table 6.3. Weighted accuracies for different optimizing approaches.

	Which Layers of Network Optimized?	
Layers/Baseline	All Layers	Last Layer
Individual (Baseline)	72.6%	72.6%
Pretrained	29.6%	29.6%
Common	37.6%	39.4%
Task-Specific	63.3%	63.8%
Personalized	72.2%	**72.4%**

Table 6.4. Weighted accuracies for various federated averaging algorithms.

	Federated Averaging Algorithm				
Layers/Baseline	FedAVG	FedProx u=0.1	FedProx u=0.3	FedProx u=0.6	FedYogi
Individual	72.6%	72.6%	72.6%	72.6%	72.6%
Pretrained	29.6%	29.6%	29.6%	29.6%	29.6%
Common	39.4%	38.6%	36.8%	40.9%	34.6%
Task-Specific	63.8%	64.0%	64.1%	64.1%	47.5%
Personalized	**72.4%**	72.2%	72.3%	72.2%	66.4%

Table 6.5. Weighted accuracies for various layer sizes.

	Layer Size				
Layers/Baseline	Default	Pretrain.=8	Common=8	Task-Sp.=8	Personal.=8
Individual (Baseline)	72.6%	72.6%	72.6%	72.6%	72.6%
Pretrained	29.6%	29.6%	29.6%	29.6%	29.6%
Common	39.4%	41.5%	39.8%	42.5%	38.4%
Task-Specific	63.8%	60.2%	58.7%	63.8%	63.4%
Personalized	**72.4%**	69.7%	67.4%	72.2%	71.6%

Table 6.6. Weighted accuracies for various division types of dataset 9.

	Division Type		
Layers	Random Division	Ordered Division	Division by User-ID
Pretrained	34.0%	16.7%	31.7%
Common	34.5%	38.5%	37.4%
Task-Specific	74.0%	47.2%	63.2%
Personalized	74.1%	47.4%	**75.3%**

7. CONCLUSION

This thesis introduces FedOpenHAR, which utilizes federated multi-task transfer learning for human activity classification and device position identification tasks with motion sensor data. We compare its performance with various models of centralized learning and federated learning combined with the DeepConvLSTM-based architecture. Models are trained using the integrated OpenHAR dataset containing ten smaller datasets. In multi-task federated transfer learning, the obtained success rates (72.4%) are similar to the baseline's success (individual training - 72.6%) and are better than a fully centralized approach (64.5%).

FedOpenHAR is evaluated with different parameters, such as different numbers of pretraining epochs, personalization epochs, layer sizes, client sampling percentages and various training percentages. In addition to these parameters, different federated averaging algorithms and distributing the clients by users in a single dataset are explored. Both pretraining and personalization epoch variations had a minor effect on FedOpenHAR's overall performance, and the best distribution of layer sizes is found as a balanced (4-4-4-2) distribution. FedOpenHAR's success slightly drops as client sampling and training percentages drop, but the drop becomes quite noticeable after a certain percentage. FedAVG worked the best among all of the federation algorithms. As it is based on simple weights aggregation, our deep layered hierarchy benefited most from the simple yet effective averaging algorithm of FedAVG. Lastly, distributing the clients by user-id in a single dataset even increased the overall success, as users are a natural distribution for the clients, which makes it more suitable for classification.

At the very end, we achieved the goal of implementing human activity recognition with a federated multi-task transfer learning architecture, first in the domain.

During the experiments in this thesis, many advancements have been explored to improve FedOpenHAR's performance. However, the default settings and parameters

are the best fit in many of the approaches. The important question is, why are the differences with the defaults so small that they can even be considered negligible? The response to the latter would be that FedOpenHAR is a multiple-layered architecture, and each layer consists of multiple training epochs of a multiple-layered DeepConvLSTM network. This being said, it is quite understandable that a change of a specific parameter does not affect much the endpoint (overall success) of our algorithm, as many of the parameters only affect a unique portion of the algorithm that we are not able to see the individual effect on end results. Even though the defaults fit well, there are still possible situations to consider further. Client selection by size and simulation of random client drops or late clients and their possible effects on overall success considering other parameters such as latency and resource usage are possible aspects to assess in further studies.

REFERENCES

1. Kumar, P., S. Chauhan and L. K. Awasthi, "Human Activity Recognition (HAR) Using Deep Learning: Review, Methodologies, Progress and Future Research Directions", *Archives of Computational Methods in Engineering*, Vol. 31, No. 1, pp. 179–219, 2024.

2. Truong, N., K. Sun, S. Wang, F. Guitton and Y. Guo, "Privacy Preservation in Federated Learning: An Insightful Survey from the GDPR Perspective", arXiv:2011.05411, 2021.

3. McMahan, H. B., E. Moore, D. Ramage, S. Hampson and B. A. y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data", arXiv:1602.05629, 2017.

4. Ordóñez, F. J. and D. Roggen, "Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition", *Sensors*, Vol. 16, No. 1, p. 115, 2016.

5. Siirtola, P., H. Koskimäki and J. Röning, "OpenHAR: A Matlab Toolbox for Easy Access to Publicly Open Human Activity Data Sets", *International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*, pp. 1396–1403, New York, USA, 2018.

6. Keçeci, C., M. Shaqfeh, H. Mbayed and E. Serpedin, "Multi-Task and Transfer Learning for Federated Learning Applications", arXiv:2207.08147, 2022.

7. Lim, W. Y. B., N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato and C. Miao, "Federated Learning in Mobile Edge Networks: A Comprehensive Survey", arXiv:arXiv:1909.11875, 2020.

8. Yang, Q., Y. Liu, T. Chen and Y. Tong, "Federated Machine Learning: Concept and Applications", arXiv:arXiv:1902.04885, 2019.

9. Ek, S., F. Portet, P. Lalanda and G. Vega, "Evaluation of Federated Learning Aggregation Algorithms: Application to Human Activity Recognition", *International Joint Conference on Pervasive and Ubiquitous Computing and International Symposium on Wearable Computers*, pp. 638–643, New York, USA, 2020.

10. Chen, Y., X. Qin, J. Wang, C. Yu and W. Gao, "FedHealth: A Federated Transfer Learning Framework for Wearable Healthcare", *IEEE Intelligent Systems*, Vol. 35, No. 4, pp. 83–93, 2020.

11. Anguita, D., A. Ghio, L. Oneto, X. Parra Perez and J. L. Reyes Ortiz, "A Public Domain Dataset for Human Activity Recognition Using Smartphones", *International European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, pp. 437–442, Bruges, Belgium, 2013.

12. Kalabakov, S., B. Jovanovski, D. Denkovski, V. Rakovic, B. Pfitzner, O. Konak, B. Arnrich and H. Gjoreski, "Federated Learning for Activity Recognition: A System Level Perspective", *IEEE Access*, Vol. 11, No. 1, pp. 64442–64457, 2023.

13. Arivazhagan, M. G., V. Aggarwal, A. K. Singh and S. Choudhary, "Federated Learning with Personalization Layers", arXiv:1912.00818, 2019.

14. Wu, Q., K. He and X. Chen, "Personalized Federated Learning for Intelligent IoT Applications: A Cloud-Edge Based Framework", *Open Journal of the Computer Society*, Vol. 1, No. 1, pp. 35–44, 2020.

15. Tan, A. Z., H. Yu, L. Cui and Q. Yang, "Towards Personalized Federated Learning", *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 34, No. 12, pp. 9587–9603, 2023.

16. Presotto, R., G. Civitarese and C. Bettini, "Semi-Supervised and Personalized Federated Activity Recognition Based on Active Learning and Label Propagation", *Personal and Ubiquitous Computing*, Vol. 26, No. 5, pp. 1–18, 2022.

17. Ouyang, X., Z. Xie, J. Zhou, J. Huang and G. Xing, "ClusterFL: A Similarity-Aware Federated Learning System for Human Activity Recognition", *International Conference on Mobile Systems, Applications, and Services*, pp. 54—66, New York, USA, 2021.

18. Presotto, R., G. Civitarese and C. Bettini, "FedCLAR: Federated Clustering for Personalized Sensor-Based Human Activity Recognition", *International Conference on Pervasive Computing and Communications*, pp. 227–236, Pisa, Italy, 2022.

19. Sandeepa, C., B. Siniarski, S. Wang and M. Liyanage, "FL-TIA: Novel Time Inference Attacks on Federated Learning", *International Conference on Trust, Security and Privacy in Computing and Communications*, p. 12, Exeter, UK, 2023.

20. Smith, V., C.-K. Chiang, M. Sanjabi and A. Talwalkar, "Federated Multi-Task Learning", arXiv:1602.05629, 2018.

21. Shen, Q., H. Feng, R. Song, D. Song and H. Xu, "Federated Meta-Learning with Attention for Diversity-Aware Human Activity Recognition", *Sensors*, Vol. 23, No. 3, p. 1083, 2023.

22. Bursa, S. O., O. D. Incel and G. Isiklar Alptekin, "Personalized and Motion-based Human Activity Recognition with Transfer Learning and Compressed Deep Learning Models", *Computers and Electrical Engineering*, Vol. 109, No. 2, p. 108777, 2023.

23. Banos, O., R. Garcia, J. A. Holgado-Terriza, M. Damas, H. Pomares, I. Rojas, A. Saez and C. Villalonga, "mHealthDroid: A Novel Framework for Agile Development of Mobile Health Applications", *Ambient Assisted Living and Daily Activities*, pp. 91–98, Granada, Spain, 2014.

24. Shoaib, M., S. Bosch, O. D. Incel, H. Scholten and P. J. Havinga, "Fusion of Smartphone Motion Sensors for Physical Activity Recognition", *Sensors*, Vol. 14, No. 6, pp. 10146–10176, 2014.

25. Siirtola, P. and J. Röning, "Recognizing Human Activities User-Independently on Smartphones Based on Accelerometer Data", *International Journal of Interactive Multimedia and Artificial Intelligence*, Vol. 1, No. 5, pp. 38–45, 2012.

26. Stisen, A., H. Blunck, S. Bhattacharya, T. S. Prentow, M. B. Kjærgaard, A. Dey, T. Sonne and M. M. Jensen, "Smart Devices Are Different: Assessing and Mitigating Mobile Sensing Heterogeneities for Activity Recognition", *ACM Conference on Embedded Networked Sensor Systems*, pp. 127–140, Seoul, South Korea, 2015.

27. Zhang, M. and A. A. Sawchuk, "USC-HAD: A Daily Activity Dataset for Ubiquitous Activity Recognition Using Wearable Sensors", *International Conference on Ubiquitous Computing, Workshop on Situation, Activity and Goal Awareness*, p. 128, Pittsburgh, USA, 2012.

28. Micucci, D., M. Mobilio and P. Napoletano, "UniMiB SHAR: A Dataset for Human Activity Recognition Using Acceleration Data from Smartphones", *Applied Sciences*, Vol. 7, No. 10, p. 1101, 2017.

29. Chereshnev, R. and A. Kertész-Farkas, "HuGaDB: Human Gait Database for Activity Recognition from Wearable Inertial Sensor Networks", arXiv:1705.08506, 2017.

30. Sztyler, T. and H. Stuckenschmidt, "On-body Localization of Wearable Devices: An Investigation of Position-aware Activity Recognition", *International Conference on Pervasive Computing and Communications*, pp. 1–9, Sydney, Australia, 2016.

31. Vavoulas, G., C. Chatzaki, T. Malliotakis, M. Pediaditis and M. Tsiknakis, "The MobiAct Dataset: Recognition of Activities of Daily Living using Smartphones.", *International Conference on Information and Communication Technologies for Ageing Well and e-Health*, pp. 143–151, Rome, Italy, 2016.

32. Bursa, S. O., O. D. Incel and G. Isiklar Alptekin, "Building Lightweight Deep Learning Models with TensorFlow Lite for Human Activity Recognition on Mobile Devices", *Annals of Telecommunications*, Vol. 78, No. 11, pp. 687–702, 2023.

33. Li, T., A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar and V. Smith, "Federated Optimization in Heterogeneous Networks", arXiv:1812.06127, 2020.

34. Reddi, S., Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečný, S. Kumar and H. B. McMahan, "Adaptive Federated Optimization", arXiv:2003.00295, 2021.

35. OpenMined, "PySyft - Perform Data Science on Data That Remains in Someone Else's Server", https://github.com/OpenMined/PySyft, accessed on Jan 8, 2024.

36. PFL, "Paddle Federated Learning", https://paddlefl.readthedocs.io/en/stable/, accessed on Jan 8, 2024.

37. TensorFlow, "TensorFlow Federated: Machine Learning on Decentralized Data", https://www.tensorflow.org/federated, accessed on Jan 8, 2024.

38. Flower, "A Friendly Federated Learning Framework", https://flower.dev, accessed on Jan 8, 2024.

39. Inovex, "Federated Learning: Frameworks for Decentralized Private Training – Part 2", https://www.inovex.de/de/blog/federated-learning-frameworks-part-2/, accessed on Jan 8, 2024.

40. Beutel, D. J., T. Topal, A. Mathur, X. Qiu, J. Fernandez-Marques, Y. Gao, L. Sani, K. H. Li, T. Parcollet and P. P. B. de Gusmão, "Flower: A Friendly Federated Learning Research Framework", arXiv:2007.14390, 2020.

41. TMP, "Mechanisms of Action (MoA) Prediction - 2nd Place Solution - with 1D-CNN", https://www.kaggle.com/c/lish-moa/discussion/202256, accessed on Jan 8, 2024.

42. Murahari, V. S. and T. Plötz, "On Attention Models for Human Activity Recognition", *International Symposium on Wearable Computers*, pp. 100–103, New York, USA, 2018.

43. Bock, M., A. Hölzemann, M. Moeller and K. Van Laerhoven, "Improving Deep Learning for HAR with Shallow LSTMs", *International Symposium on Wearable Computers*, pp. 7–12, New York, USA, 2021.