2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications

Human Activity Recognition Using Federated Learning

Konstantin Sozinov
KTH Royal Institute of Technology
Stockholm, Sweden
sozinov@kth.se

Vladimir Vlassov
KTH Royal Institute of Technology
Stockholm, Sweden
vladv@kth.se

Sarunas Girdzijauskas
KTH Royal Institute of Technology
Stockholm, Sweden
sarunasg@kth.se

***Abstract*—State-of-the-art deep learning models for human activity recognition use large amount of sensor data to achieve high accuracy. However, training of such models in a data center using data collected from smart devices leads to high communication costs and possible privacy infringement. In order to mitigate aforementioned issues, federated learning can be employed to train a generic classifier by combining multiple local models trained on data originating from multiple clients. In this work we evaluate federated learning to train a human activity recognition classifier and compare its performance to centralized learning by building two models, namely a deep neural network and a softmax regression trained on both synthetic and real-world datasets. We study communication costs as well as the influence of erroneous clients with corrupted data in federated learning setting.**

We have found that federated learning for the task of human activity recognition is capable of producing models with slightly worse, but acceptable, accuracy compared to centralized models. In our experiments federated learning achieved an accuracy of up to 89 % compared to 93 % in centralized training for the deep neural network. The global model trained with federated learning on skewed datasets achieves accuracy comparable to centralized learning. Furthermore, we identified an important issue of clients with corrupted data and proposed a federated learning algorithm that identifies and rejects erroneous clients. Lastly, we have identified a trade-off between communication cost and the complexity of a model. We show that more complex models such as deep neural network require more communication in federated learning settings for human activity recognition compared to less complex models, such as multinomial logistic regression.

***Index Terms*—Federated Learning, Human Activity Recognition, Privacy, Distributed Machine Learning**

I. Introduction

Human activity recognition (HAR) is a classification machine learning task where the goal is to learn which activity is performed by a certain person in a given period of time. Activities can be of different kinds, for example: sitting, standing, walking, running, biking or driving a vehicle. A HAR classifier can be suitable for various types of applications ranging from healthcare and fitness applications, for example a Fitbit [1] watch counting steps, to context aware applications like the "Do Not Disturb while Driving" feature on iOS version 11 [2]. To recognize activities, a machine learning model is trained on accelerometer and gyroscope sensor data from smart devices, smartphones or smartwatches.

A wide range of mobile applications and smart devices allows to collect huge amounts of sensor data and push progress in HAR research. Using deep learning, researchers achieve high accuracy on HAR tasks, based on the data from smart devices [3]–[5]. Yao et al. [3] propose a state-of-art deep model for mobile sensing based on accelerometer and gyroscope sensor data. The model consists of a convolutional neural network (CNN) combined with a recurrent neural network (RNN) and is trained on up to an order of GB of sensor data from smart devices. Training this model on real world data collected from smart devices leads to various implications. Mobile clients need to send a lot of data to a centralized server or a cluster for both training and inference. This is challenging due to users' billing plan, users' privacy and labeling of sensor data.

McMahan et al. [6] proposed to address the aforementioned challenges by training a classifier using federated learning. Federated learning allows to build a global model while keeping the sensor data close to the user, ensuring that the data for training and inference does not leave the user's device. This also allows preserving privacy and reducing communication cost when training and using the model. Since a lot of motion data for training and inference needs to be collected to classify activities with high accuracy, as in the model built in in [3], training the classifier using federated learning algorithm fits the problem naturally. However, the algorithm is not so much tested in practice, and there are open questions of the implementation details such as what are the limits of the algorithm and in what environments it is suited.

In this study we evaluate performance of the federated learning and show that federated learning can be used instead of centralized learning for training a HAR classifier. Our baseline is a HAR classifier trained using centralized learning, i.e., a classifier trained on sensor data collected and stored on a central server, using Stochastic Gradient Descent (SGD). We compare HAR classifier trained using centralized learning with HAR classifier trained using federated learning for three data distributions among clients: (1) unbalanced and non-independent non-identically distributed (non-IID) data, (2) uniformly distributed data, and (3) skewed data. In the case of non-IID data, federated clients have at most 2 activities from the dataset, where one of the activities has 50 % less data points than another. In the case of uniformly distributed data, clients have all activities from the dataset and the activities are uniformly distributed across the clients. In the case of skewed data, the majority of clients have uniformly distributed data and one client has a single data class (*a client with skewed distribution*). We try to map the distributions to real world scenarios, where clients can perform different activities, or some clients may have an activity that is less frequent, for example biking. For the case of skewed data, we investigate the local test accuracy of a client with skewed distribution. Furthermore, since in federated learning the data is distributed among many clients, we study how erroneous clients, can affect federated learning and how it can be mitigated. We define *an erroneous client* as a client who produces sensor data which is completely outside of the distribution of sensor data coming from normal clients. Both centralized and federated methods are evaluated using two different models: (1) a multinomial logistic regression, often called a softmax regression, and (2) a deep neural network (DNN).

The main contributions of the paper are as follows. First, we have built two models for the task of HAR, namely, a softmax regression and a deep neural network. Each of the models has been trained in two ways, using federated learning as a novel distributed machine learning approach, and using centralized learning, on a HAR dataset with three different data distributions among clients, non-IID data, uniformly distributed data, and skewed data. Second, we have evaluated performance and communication cost of both HAR models, softmax regression and DNN, trained with federated learning compared to the models trained with the centralized learning for different data distributions; and we have shown that federated learning allows building models for HAR with an acceptable accuracy comparable to the centralized learning, while preserving data privacy and significantly reducing communication cost for simpler models. Finally, we we have studied the effect of corrupted data from erroneous clients on the performance of federated learning and proposed a federated learning algorithm with detection and rejection of erroneous clients that improves performance of the models for human activity recognition using federated learning.

The rest of the paper is structured as follows. Section II describes the background. Section III introduces the models for HAR: a softmax regression and a DNN. Section IV presents the experimental and evaluation setup, including dataset, tools and frameworks used in this study, the federated learning algorithms, and evaluation metrics. Section V provides the experimental results. Section VI discusses related work. We conclude and highlight our future work in Section VII.

II. Background

The classification task of a HAR classifier, $\mathcal{C}$, is to predict different human activities based on a set of body sensors, $\mathcal{S}$. Typically $\mathcal{S}$ consists of triaxial accelerometer and/or gyroscope sensors, call them A_x, A_y, A_z and G_x, G_y, G_z. Beyond accelerometer and gyroscope magnetic field sensors can be used, but they are less common [7]. An accelerometer is a hardware unit that measures acceleration forces, typically in $\frac{m}{s^2}$, while

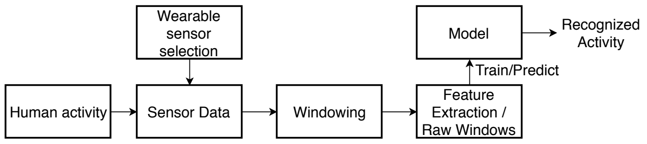

Fig. 1. Overview of a HAR classifier

gyroscope measures the rotation of a given device, for example a mobile phone, in $\frac{rad}{s}$. Almost all modern smartphones and smartwatches have accelerometer and gyroscope built in into them. The sensors are used for different use cases, the gyroscope for performing screen rotation and the accelerometer for counting steps or augmented reality applications.

Before the emergence of deep learning as the state-of-the-art on HAR tasks, researchers in the area of human activity recognition used various types of hand-crafted features for training a HAR classifier. The feature engineering was an important process and depended a lot on the target classification task. In general, most of the hand-crafted features based on sensor data fall into three different categories: time domain features, frequency domain features and time-frequency analysis. A wide range of "classical" machine learning models were used for training a HAR classifier before the deep models came: unsupervised methods such as k-means, probabilistic methods as Naive Bayes, Support Vector Machines (SVM), Perceptron and other [8].

In contrast to hand-crafted features and classical machine learning models, deep learning exploits benefits of having huge amount of data and deep nonlinear models. When deep models are used, the feature extraction process can be simply omitted. In most cases, raw windowed sensor data is fed directly to the classifier. Long short-term memory recurrent neural networks (LTSM RNN), CNNs or a combination of both models is a dominant approach in HAR today [3], [9], [10].

Figure 1 summarizes training and inference of a HAR classifier. The diagram does not cover practical aspects of using a HAR classifier such as how the data is collected, how the model is distributed to the end users or how the data is labeled. First, sensor data is collected from mobile devices with accelerometer and gyroscope, second, depending on a model selection for $\mathcal{C}$, windowed data is used for feature extraction or fed directly to the classifier for the training. At prediction time, data is collected using the same window length and then, again depending on the model selected for a HAR classifier, features are extracted or raw windowed data is fed into $\mathcal{C}$ which predicts the target label, such as walking.

A. Federated Learning

In federated learning the learning task is done by a federation of participating devices, which produce the training data, instead of centralizing all training data on a server or in a cloud. The initial algorithm is proposed by [6] and focuses on training using SGD and modeling using different neural networks, in particular DNNs, CNNs or RNNs. The main objective of the federated learning algorithm is to learn a model $\mathbf{M}$ by using a subset of clients u from a set of all clients P. Each client has its own local dataset $\mathbf{D}_k^r$ at round r, a local model $\mathbf{H}_k^r$, and performs one or more iterations of SGD. After several iterations of SGD are performed by a client, it sends the next local model $\mathbf{H}_k^{r+1}$ to the server which holds the global model. A synchronous update of the global model is used. When all clients are done with several epochs of the SGD, the server collects all local models and updates the global model $\mathbf{M}_{r+1}$. Authors evaluate the global update using two different approaches, federated SGD, where clients update the local model only once and federated averaging where clients update $\mathbf{H}$ multiple times. Federated averaging outperformed federated SGD with respect to number of communication rounds needed to achieve the target accuracy and is computed using the following equations [6]:

$$\mathbf{H}_k^{r+1} = \mathbf{H}_k^r - \eta \mathbf{g}_k^r; \quad \mathbf{M}_{r+1} = \sum_{i \in D_k} \frac{n_k}{n} \mathbf{H}_k^{r+1} \qquad (1)$$

where $\mathbf{H}_k^{r+1}$ is the local update of the local model, $\mathbf{g}_k$ are the gradients computed using backpropagation, $\mathbf{M}_{r+1}$ is the next global model, η is a learning rate, n is the sum of all data points and n_k is the number of local data points.

The federated learning algorithm has several advantages over training a classifier using a centralized approach. One of the biggest advantages is the fact that sensitive data produced by a client is kept on the client's device. This allows to train models using sensitive client data such as URLs, password or keyboard strokes. Moreover, it allows to reduce communication costs if data cannot be logged to a central server. Two data distributions differentiate federated learning from centralized or distributed optimization: non-IID and unbalanced data. The algorithm itself tries to adapt for non-IID and unbalanced data by weighting each local model by a number of data points the local model was trained on, $\frac{n_k}{n}$ in Equation 1. Another aspect of federated learning is the synchronous update, where clients who are slow, not responding or offline affect the learning.

The federated learning algorithm proposed by [6] showed that communication costs using federated averaging can be reduced by a factor of 10-100 compared to federated SGD. Konen et al. [11] came up with several techniques for reducing the communication costs even more. The authors propose two types of updating local client's model before communicating it to a central server: structured and sketched updates. Using a structured update the client maps the original local model $\mathbf{H}_k^r$ to a lower dimensional space. Using a sketched update the client compresses $\mathbf{H}_k^r$ by using, for example, a probabilistic quantization. Using CNNs and LSTMs authors show that communication costs can be reduced by two orders of magnitude compared to the original federated learning algorithm.

III. Models

In this study we build two different models, a DNN and a softmax regression model. We use a DNN model rather than state-of-the-art models, such as CNNs or RNNs, because of two main reasons: technical limitations of the implementation, computational and communication costs of federated clients. A CNN model contains more parameters compared to the DNN, because of the convolutional kernels used in a CNN architecture before the feedforward layer, used to extract features. A RNN model is more computationally expensive compared to the DNN or softmax regression model, backpropagation algorithm for computing the error trough many layers is more computationally expensive compared to backprop in the DNN.

Three aspects were taken into account while designing the models: (1) the number of parameters of each model has to be small since clients are limited in terms of computational power; (2) the number of parameters also effects the amount of information transmitted between the server and clients in federated learning and (3) we do not aim to achieve best accuracy on the task of HAR, but rather aim to show that we can achieve similar accuracy, compared to centralized models, while data is kept on a client instead of collecting a lot of data and training a centralized classifier. Thus, in centralized training, we perform a grid search [12] for different parameters, where we seek for a best combination of hyperparameters, such as architecture, learning rate and batch size. For softmax regression we explore different number of coefficients in the model by producing polynomial features and for DNN we explore different number of hidden layers and neurons in each layer. Both models are trained using minibatch SGD on the 80 % of data from smartphones from the dataset and form our baseline: HAR models trained using a centralized approach trained on all available data. The performance of the baseline models is then evaluated on the 20 % test data of the dataset.

IV. Evaluation

A. Dataset

The Heterogeneity Human Activity Recognition Dataset produced by [13] is used in this project. In the study, authors show the heterogeneity of sensor data from different smartphones and smartwatches. The authors claim that when a HAR classifier is deployed to multiple mobile devices, it often performs significantly worse compared to results reported in the research. This is due to variety of sensor hardware, operating systems and different mobile models. The dataset contains the following human activities: biking, sitting, standing, walking, stair up and stair down and null. Null is used when the activity lacks the ground truth annotation. In our experiments sensor data without the label is deleted from the dataset. The dataset has a uniform distribution between different activities as well as activities performed by each user. The data was recorded using nine data producers who carried 8 different Android smartphones and 2 Android smartwatches. The data was gathered as fast as mobile sensors could provide it, with sampling frequency between $50Hz$ and $200Hz$. While producing the sensor data, the smartphones were kept around the waist and smartwatches were worn on each arm.

B. Data Preparation

We construct a feature dataset $\mathbf{D}$ from Heterogeneity Human Activity Recognition Dataset for training the softmax regression and the DNN models. In $\mathbf{D}$ we extract statistical features from windowed accelerometer and gyroscope values from each user in the dataset. We apply sliding windowing over 3 seconds with 50 % overlap and extract mean, standard deviation, maximum and minimum features from each window. Since the data is collected from different smartphones with different sampling rates, it is preprocessed to a fixed sampling rate of $65Hz$ on each user's dataset. Concretely, if one window of 3 second contains more data points than a window sampled with $65Hz$ (200 measurements), we draw points uniformly at random from the window. If the window contains less data points we use interpolation to fill the missing values to reach $65Hz$ sampling frequency. We calculate magnitude for both accelerometer and gyroscope and extract same statistical features from it as we do from $A_x, A_y, A_z, G_x, G_y, G_z$. The resulting dataset for training the models has 32 features.

We simulate clients with outlying data from $\mathbf{D}$ and study how it affects federated learning. Given that $\mathbf{D}$ contains 9 users, we simulate up to 9 erroneous users by taking random uniform points from each real user feature's quantile. Two different ranges are used from which random observations are derived, first range is between 1 % and 5 % quantile of a real user feature and the second is between 95 % and 99 % quantile of same feature. We perform this process for all features for one user's dataset and assign the same labels as in the real user's dataset. Thus, the resulting generated dataset for an erroneous client contains noise as on data level, since data is generated on tails of a real user's dataset distribution as well as noise on the activity (label) level, because data is generated independently from each activity.

C. Data Partitioning and Distribution

We divide $\mathbf{D}$ into 80 % training data and 20 % test data in centralized learning, and refer to each of them as global train dataset and global test dataset. The global test data is used both in centralized training and in federated learning, for studying the accuracy. The global train dataset is used for studying the convergence. For distributing $\mathbf{D}$ over clients in federated learning, we use three different methods: unbalanced and non-IID, uniformly distributed and skewed.

For simulating unbalanced and non-IID distributions over the clients in federated learning, we group the data by each user in the dataset and split into 3 different generated clients in federated learning. Since each user perform 6 activities, and activity sensor data per user is evenly distributed, we produce pathological non-IID data for clients in federated learning by taking 2 activities from each user and assigning them to one client. Hence, each client in federated learning will get 2 different activities from one user. After this we select one activity on each client to have 50 % less data points in order to achieve unbalanced data. In uniformly distributed data partitioning, we simply group the data by a user and let each client in federated learning obtain data from one user in the dataset. For studying the local performance on clients with skewed distributions, we partition the data uniformly from all users in the dataset to federated clients, expect one user, on which we take only one activity. For studying how clients with corrupt data affect federated learning, we partition data from real users uniformly across clients in federated learning and then we add one erroneous client per user in each experiment until we reach the same number of erroneous clients as real clients. Each local dataset is divided into 90 % training data and 10 % test data. In training data on each client, we exclude data points which are same as in global test dataset.

D. Federated Learning with Spark, Dataproc and TensorFlow

We built a prototype implementation of the federated learning algorithm for HAR upon the MapReduce algorithm using Apache Spark and Dataproc. The prototype consists of one master node and several client workers. The master node is in charge of the whole workflow of the federated learning algorithm and delegates training tasks to the client workers. Each training task is a TensorFlow graph of the selected model, the DNN or the softmax regression, and executed on each client's sensor local dataset. The master node assigns each local training dataset to each client worker. Once a client worker gets the dataset, the federated learning algorithm starts. Each worker will train its local model for a number of epochs on the local dataset using TensorFlow, update the local model and send the model back to the master node. When all clients performed e number of epochs, the master node updates the global model and sends it again to the client workers. The process continues in r number of communication rounds. Algorithm 1 defines the whole implementation of the federated learning training using Spark, Dataproc and TensorFlow. Lines 3-5 are executed on each client worker and corresponds to a *map* operation in Apache Spark. The rest of the algorithm is executed on the master node, where $collect$ is an operation from Spark and $train$ is a TensorFlow training procedure.

In order to simulate mobile hardware, we choose smallest instance available on Dataproc as workers, *1-standard-1* with 1 virtual CPU 2.0 GHz and 3.75 GB RAM. As master node we choose *1-highmem-8* with 8 virtual CPUs 2.6 GHz and 52 GB of memory. Moreover, we limit workers memory to 1 GB in the Spark configuration and each worker will have only one thread per CPU. This hardware setup allows to come near what today's modern smartphones provide, in terms of computing resources and memory.

E. Rejection of Erroneous Clients in Federated Learning

To address the problem of erroneous clients in the federated learning, we propose a rejection algorithm based on the test accuracy of the each individual client in the federated learning. After a number of communication rounds, or so called *cutoff round*, we evaluate all local models using each client's local dataset. Then, using the test accuracy on each of the local test datasets and an accuracy threshold t, we reject clients that have the test accuracy below the threshold. For selecting t, we investigate local test accuracy on a normal client and compare it to the local test accuracy on an erroneous client.

Algorithm 1 Federated Learning Algorithm for Training a HAR Classifier

```
Input: $\mathbf{M}_i$ global model on a given round, $\mathbf{H}_k^i$ - local model
    on each client, $n$ - number of sensor data observations
    across all clients, $n_k$ - number of observations on each
    client, $L$ - set of local datasets for training, $r$ - number of
    rounds in the federated learning algorithm, $e$ - number of
    training epochs per one round, $b$ - batch size of training
    data, $K$ - set of clients participating in federated learning.
1: while $r \neq 0$ do
2:     for all l $\in L$ in parallel do
3:         $\mathbf{H}_k^i = \mathbf{M}_i$
4:         $\mathbf{H}_k^{i+1}$ = train($\mathbf{H}_k^i$, l, e, b)
5:         send($\mathbf{H}_k^{i+1}$)
6:     end for
7:     collect()
8:     $\mathbf{M}_{i+1} = \sum_{k \in K} \frac{n_k}{n} \mathbf{H}_k^{i+1}$
9: end while
```

Algorithm 2 Federated Learning Algorithm for Training a HAR classifier with Rejection of Erroneous Clients

```
Input: Same as in Algorithm 1 and cutoff_round - round
    when to start reject clients, $t$ - test accuracy threshold
    for rejecting a client, $H$ - set of all local models from all
    clients.
1: while $r \neq 0$ do
2:     for all l $\in L$ in parallel do
3:         $\mathbf{H}_k^i = \mathbf{M}_i$
4:         $\mathbf{H}_k^{i+1}$ = train($\mathbf{H}_k^i$, l, e, b)
5:         send($\mathbf{H}_k^{i+1}$)
6:     end for
7:     $H$ = collect()
8:     if $r$ == cutoff_round then
9:         $H$ = reject($H$)
10:    end if
11:    $\mathbf{M}_{i+1} = \sum_{k \in K} \frac{n_k}{n} \mathbf{H}_k^{i+1}$
12: end while
13:
14: function REJECT($H$)
15:     for all $\mathbf{H}_k \in H$ do
16:         $acc_k$ = $\mathbf{H}_{k\_local\_acc}$
17:         if $acc_k < t$ then
18:             $H$ = $H \setminus \mathbf{H}_k$
19:         end if
20:     end for
21:     return $H$
22: end function
```

Algorithm 2 describes the federated learning algorithm for HAR accounting for erroneous clients. Function *reject* is iterating on all local test accuracies received from clients and is executed on the master node. When the cutoff round is reached, lines 8-10 are removing clients that considered to be erroneous from the federated averaging of all local models.

F. Evaluation Metrics

For the evaluation of the proposed method we track three different metrics, value of the objective function of the minimization, often called loss value or cost value, an accuracy on test data and data upload in federated learning. For training the DNN, ℓ is the cross-entropy between the observations and the predicted label, and for training a softmax regression ℓ is a logistic loss function. Accuracy metric is a fraction between correct predictions and total number of observations.

For our baseline models we observe the mean test set accuracy during the training. We put this in contrast with convergence and test accuracy of the models trained with federated learning, to get a sense whenever the target function converges or not, and what accuracy we can achieve using federated learning. We examine the steepness of the loss function and test accuracy on number of communication rounds in the federated learning. We assume that one communication round is when clients send local models to global server and receive back the updated model. For the federated learning the value of the loss function will be calculated on the training data distributed across clients, using the global model after the new global model is produced. The accuracy metric in the federated learning will be calculated using the same approach but on the global test dataset. We point out that we use the same global train and test datasets in the federated learning as we use in the training of the baseline models, but in federated learning the global train dataset is distributed across clients. For studying the communication efficiency of the federated learning we track loss and test accuracy on the data upload from each client in megabytes and compare it to the size of each client's local dataset. Data upload from each federated client can be multiplied by 2, in order to get a sense of how much data is transmitted in both uploading the local model and downloading the new global model in one communication round between the clients and the central server.

V. RESULTS

In this section we present the results of using federated learning for the task of HAR. We compare centralized SGD training on a central server where all sensor data is available to the federated learning with three different data distributions across clients in federated learning. We evaluate federated learning for HAR in terms of performance for clients with skewed distributions and erroneous clients. We also present communication and computational costs for federated learning.

A. Centralized SGD Training

For a comparison between models trained using SGD on all data and federated learning, we form our baseline as the best performing centralized models. Using mean validation accuracy, we identify the top 5 models of each type found during grid search of different parameters. For the softmax regression, we perform grid search through 3 parameters: degree of polynomial features (adding more parameters to the model by feature crossing), learning rate and minibatch size. For the DNN, we seek through number of hidden layers in

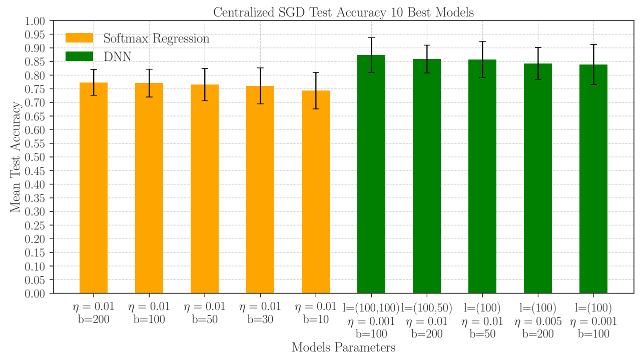

Fig. 2. 5 best centralized models of each type found with grid search of different parameters. Here η is the learning rate, b is minibatch size and l is number of hidden layers in the DNN and neurons in each layer, e.g. $l = (100, 100)$ is a network with 2 hidden layers 100 neurons in each.

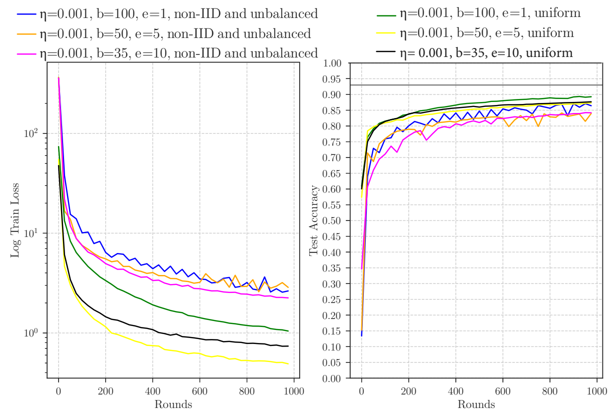

Fig. 3. A DNN HAR classifier trained using federated learning with two different data distributions among participating clients. The grey line indicates best centralized accuracy achived using the DNN model.

the architecture and neurons in each layer, learning rate and minibatch size.

Figure 2 depicts test accuracy of the 5 best models of each type found during the grid search and corresponding parameters. Each of the models of each type performed similarly, without any bigger uplift in test accuracy. The best performing DNN model consist of 2 hidden layers and 100 neurons in each layer, with a total of 24006 parameters. For the softmax regression model polynomial feature crossing did not add any boost in the accuracy and made the model diverge. Thus, the top 5 regression models are using only the original features from the dataset and contain 198 parameters. By using early stopping while training the two best models, best test accuracy for the DNN was 93% and 83% for the softmax regression.

B. *Federated Learning for HAR*

Using uniform and non-IID data distributions we investigate how federated learning is performing when data on a federated client is based on the usage of a mobile device. Figure 3 shows the convergence and the test accuracy of DNN and softmax regression models trained for HAR classification. We observe that federated learning can compete with models trained with centralized learning. Federated learning can be seen as an optimization of two different target functions, $F(\mathbf{M})$ which aims to build a global model $\mathbf{M}$, by federated averaging all local models $\mathbf{H}_k$ and $f_k(\mathbf{H})$, which aims to optimize for local data on a given client trough SGD. As we try to optimize local models for non-IID and unbalanced data distributions, $f_k(\mathbf{H})$ is a poor approximation of $F(\mathbf{M})$, since client's data is not representative of the whole distribution. This data distribution can be compared to uniformly distributed data across clients in federated learning, where optimizing $f_k(\mathbf{H})$ can be seen as optimizing $F(\mathbf{M})$. This can be observed on the results we get during the study, where a more realistic scenario with non-IID and unbalanced data distribution among clients, showed an accuracy of 87% compared to 93% in centralized SGD training, see Table I. On the other hand, observing the trend line of the test accuracy for the DNN in Figure 3, we see that it still has a positive trend, which hypothetically means that if we train for more communication rounds, we improve the accuracy of the federated learning models for both non-IID and uniform data distributions. This implies that more data needs to be send to a central server, as well as more computational power is required on the mobile devices of federated learning clients. In case of uniformly distributed data among clients, the difference between centralized SGD and federated learning is even less significant, however this scenario is less common in the real world applications, since data on each federated client might be skewed towards specific human activities.

Federated learning was proposed for problems where data is privacy sensitive and hard to collect from numerous mobile clients because of the communication costs. At what cost do the benefits of the federated learning come while training a HAR classifier? In this study, we applied feature engineering to produce statistical features from the given dataset for HAR, while in state-of-the-art models, such as DeepSense model, raw sensor data, which consists of more observations, is used directly to train a centralized classifier. In order to achieve the best test accuracy using the DNN model in federated learning on unbalanced and non-IID data with $b = 100$ and $e = 1$, we need up to 1000 communication rounds, which means 1000 epochs trough local dataset on each client and model exchanges with the global server, compared to only 7 epochs of training in the centralized SGD model. Furthermore, observing the results from Table I, we see that, in this specific case, using a more complex model as DNN requires sending up to two order of magnitude more data containing local models (and thus incurring higher computational costs at the clients) in order to achieve high accuracy, compared to the size of the local dataset used to produce the local models. However, the model weights do not directly depend on the existing raw data, but on the complexity of the user behaviour phenomenon that we are learning and extracting from the data. Thus, it is expected that in case of wide range of applications, such as image or speech recognition, the size of the local dataset would be much larger compared to datasets used in HAR. On the other hand, as Table I shows, less complex models such as softmax regression allows to significantly save communication costs in federated learning. Thus, using less complex models can be a viable solution for most of the real-world applications since user behavior phenomenon, observed in HAR, are often simple enough to be captured by relatively simple models.

As we compare different models used in this study, we see a trade-off between having a less complex model, which requires less computational resources and data upload, but produces a global model with lower accuracy, and a more complex model which requires much more computational and communication resources. This trade-off indicates that federated learning requires less communication and compute resources when using less complex models, but at the cost of a lower accuracy.

C. Study of Clients With Skewed Distributions

In this experiment we study how well the global model fits a client who has data distribution skewed towards one activity. We select one client to perform only one activity, whereas other clients perform all activities uniformly distributed among them. Since the dataset contains 9 clients, we select each one of them separately to have a skewed distribution and perform the federated training with 1 client with a skewed distribution and 8 clients where the data is uniformly distributed.

The results presented in Table II indicate that the new global $\mathbf{M}^{r+1}$ is not always the best choice for clients with a skewed distribution, but it clearly depends on the performed activity and not on the federated learning algorithm. Each row in the table shows one experiment, where we let one client have only one single activity while other 8 clients have all of the activities. Test accuracy is presented for the label used in the experiment for both a client with skewed distribution towards the label and all other clients, together with the mean test accuracy across normal clients and the client with a skewed distribution. Overall, the client with skewed distribution achieved similar mean test accuracy to clients with uniformly distributed data. Concretely, walk, stairs up and stairs down activities showed the accuracy below 50%, for some of the normal clients and the accuracy of 30% for a client with skewed distribution towards stairs up. However, for other activities, such as sit or stand, the mean local test accuracy on the client with skewed distribution tend to be higher than mean local accuracy on the uniformly distributed clients. The table justifies that even for clients with uniform distributions and all labels, some activities are harder to predict than others. Activities such as walk, stairs up and stairs down get the accuracy below 50% for at least 2 clients with uniform distribution of labels, while activities such as sit or stand are easier predictable and always get the accuracy above 70%. We think that this is due to the fact that static activities such as sit or stand are easier to predict than other activities.

The results indicate that using the new global model $\mathbf{M}^{r+1}$ directly after receiving it from the central server gives a lower local test accuracy, compared to using the next produced local model after local training. That is because when a client starts to optimize for the local data distribution, it can achieve much higher local test accuracy. Concretely, a very high local test accuracy, even using simple models as softmax regression, can be observed after one epoch of learning while producing a next local model. It indicates that federated learning can be used to build more personal models instead of optimizing the global model. For instance, if the newly received global model $\mathbf{M}^{r+1}$ performs really bad in terms of local test accuracy compared to the newly produced local model $\mathbf{H}_k^{r+1}$ after local training, a similar rejection algorithm as Algorithm 2, can be applied on the client side for rejecting the newly received global model $\mathbf{M}^{r+1}$ and optimizing only its local model instead. This can allow to build better local models for clients with data distributions that differ from the whole set of federated clients, for example clients that perform an unique activity. Practically, we can say that if the local test accuracy on a client k using its local model $\mathbf{H}_k$ is much higher than the test accuracy using $\mathbf{M}^{r+1}$ by a given threshold t, the new global model $\mathbf{M}^{r+1}$ can be rejected and k can use its local data for optimizing its local model $\mathbf{H}_k$.

D. Erroneous Clients Effect on Federated learning

Since in federated learning the data is distributed across many clients, we think that clients with outlying data is an open issue of the algorithm. Observing the results from Table III, we see that having a fraction of 1 of erroneous and real clients decrease the mean test accuracy after 300 rounds of training of the DNN model for HAR from 0.82 to 0.73. By explicitly identify erroneous with the presented algorithm 2 for client rejection, we were able to achieve better mean test accuracy with a fraction of 1 erroneous and real clients as without any erroneous clients. Clearly as we use the earliest cutoff round such as 25, we are getting better results. Rejecting potential erroneous clients too early may lead to problems too: the normal clients need some rounds for achieving the accuracy above the cutoff threshold.

VI. Related Work

Ramakrishnan et al. [14] propose a cloud learning for training a HAR classifier built into a context-aware application for people with diabetes disease. The work justifies that key challenges of integrating a HAR classifier on a mobile device are resources available on the device such as battery, communication bandwidth and computational resources. The authors use a hidden markov model to recognize human activities on an Android application. The model is trained using statistical features on a windowed triaxial accelerometer data collected from mobile devices. Different sampling frequencies are used for collecting accelerometer sensor data based on the battery level of the device. Another feature proposed by the authors is smart offloading of training to the cloud, based on a Markov decision process (MDP). Taking into account resources available (CPU, memory, storage, communication) on a device, MDP models the best offloading policy given these factors. When the recognition is offloaded to the cloud,

TABLE I
OVERVIEW OF THE BEST PERFORMING FEDERATED AVERAGING MODELS TRAINED ON DIFFERENT DISTRIBUTIONS OF DATA AMONG CLIENTS.

Best Models	Parameters	Data Distribution	Federated Accuracy	Centralized Baseline	Data Upload per Client[a]	Local Dataset Size
DNN	$\eta = 0.001$, b=100, e=1	Non-IID & Unbalanced	0.87	0.93	51.29 MB	0.25 MB
DNN	$\eta = 0.001$, b=100, e=1	Uniformly Distributed	0.89	0.93	52.78 MB	1.06 MB
Softmax Regression	$\eta = 0.001$, b=50, e=5	Non-IID & Unbalanced	0.78	0.83	0.3 MB	0.25 MB
Softmax Regression	$\eta = 0.001$, b=50, e=5	Uniformly Distributed	0.80	0.83	0.03 MB	1.06 MB

[a]Communication cost to achieve best accuracy. Can be multiplied by a factor of 2 to get total data usage on a client.

TABLE II
STUDY OF LOCAL TEST ACCURACY FOR EACH OF THE LABELS FOR NORMAL CLIENTS DURING FEDERATED LEARNING, WITH CLIENTS THAT HAVE SKEWED DISTRIBUTIONS TOWARDS ONE SINGLE ACTIVITY USING THE DNN MODEL.

Experiment	Test Accuracy on Normal Clients for the Label									Skewed Test Accuracy
	a	b	c	d	f	g	h	i	μ	
Sit	0.92	0.87	0.74	0.93	0.99	0.98	0.99	0.99	0.92	0.99
Stairs down	0.62	0.70	0.58	0.96	0.64	0.86	**0.18**	0.92	0.68	0.72
Stairs up	0.93	0.91	**0.57**	**0.25**	0.72	0.96	0.76	**0.11**	0.65	**0.30**
Walk	**0.26**	**0.20**	0.96	0.84	0.95	**0.53**	0.91	0.88	0.69	0.98
Bike	0.87	0.81	0.92	0.91	0.72	0.97	0.98	0.94	0.89	0.88
Stand	0.90	0.83	0.91	0.99	0.98	0.99	0.98	0.83	0.92	0.98
μ	0.75	0.72	0.78	0.81	0.83	0.88	0.80	0.77	0.79	0.80

TABLE III
EFFECT OF HAVING ERRONEOUS CLIENTS AND REJECTION ALGORITM 2.

Model	μ_{acc} Without Outliers	μ_{acc} With Outliers	μ_{acc} Using Rejection Algorithm
DNN	0.82	0.73	0.77 - cutoff 25 0.76 - cutoff 50 0.74 - cutoff 100
Softmax Reg.	0.75	0.67	0.74 - cutoff 25 0.73 - cutoff 50 0.71 - cutoff 100

the device sends the sensor data to a cloud environment using Wi-Fi, whenever possible. When an activity is recognized on a cluster it is fed into an insulin dosage recommender engine. The authors' goal of the whole system is to achieve 90% classification accuracy on HAR task with battery budget of 15% over 24 hours and 5% of its communication bandwidth. Due to a bug on Android 2 the authors could not justify if the goal was achieved. The paper is the closest we found compared to the goal of our work. Still one difference is remarkable, since the paper does not mention the fact that accelerometer data is sensitive data and should be treated accordingly.

Smith et al. [15] build on the original federated learning algorithm proposed by [6]. The work shows how several statistical and systems design challenges in federated learning, such as non-IID data or stragglers, can be solved using federated multi-task learning. The core idea of multi-task federated learning is to fit separate models (one per task) based on a local dataset on each device, while still preserving structure between all local models, a global model. The authors prove that this approach is well-suited to handle the statistical challenges of the federated setting. Beyond this, the algorithm takes into account system challenges in synchronous federated learning. The solution allows to partially solve a local task, i.e. interrupt an iteration of training the local model, if a node is considered as a straggler. The algorithm is evaluated on different datasets, among others UCI Human Activity Recognition dataset [16]. Each producer of the sensor data in the dataset is modeled as a separate task in multi-task learning. The data is partitioned randomly into 75 % training and 25 % test set and three (multi-task local/global and centralized) SVM classifiers are then compared. On the task of HAR a multi-task global model significantly outperforms both global and local federated learning approaches.

Shokri and Shmatikov [17] propose an approach to distributed collaborative deep learning where a set of selected parameters and gradients are sent to a central server when training a classifier using SGD. This approach works for any type of neural network and preserves privacy of participant training data without sacrificing the accuracy of the resulting models. Using this approach, the authors propose a distributed collaborative training algorithm where each learner trains a model locally on its data and then exchange a subset of parameters and gradients with a central server asynchronously. In contrast to [17], the federated learning method proposed in [6] uses all parameters and synchronous updates.

VII. CONCLUSION AND FUTURE WORK

We thoroughly investigated the applicability of federated learning to the task of human activity recognition. Federated learning enables training without sending privacy sensitive raw data to a central server. We showed that federated learning for HAR is sufficiently robust under variety of workloads and produces models with acceptable accuracy comparable to centralized learning. In our experiments on synthetic and real world datasets, federated learning achieves accuracy of up to 89% compared to up to 93% in centralized learning at the price of higher communication cost for complex models with a high number of parameters, such as DNNs. Less complex models for HAR, such as softmax regression, allow reducing communication costs of federated learning, while still achieving an accuracy up to 80% compared to 83% in centralized learning. Thus, lower complexity models should be used for communication sensitive applications, while higher complexity models should be used for application that require higher accuracy. Communication cost of federated learning can be further reduced by applying compression algorithms, as proposed in [11]. We identified an important issue of clients with corrupted data and propose a federated learning algorithm that identifies and rejects erroneous clients while achieving an accuracy close to federated learning without erroneous clients.

A. Future Work

Although the relatively small scale experiments conducted in this work show promising results in applicability of federated learning for the task of HAR, our future work includes running experiments on much larger scale using real-world workloads that, we expect, will further confirm the viability of federated learning for HAR. For example, the next step in this study is to implement the federated learning algorithm for HAR in a mobile application with many mobile users, for instance a fitness application that can predict a current activity of a user. By training and integrating a federated model on the application, a broader number of metrics can be tracked for evaluating the federated learning as a method of training. The application would allow to benchmark metrics such as battery usage on the device for training, measure performance of the classifier using user interaction, an effect of communication problems and stragglers, and other metrics that we considered in this study such as convergence, data upload and accuracy on local test data. The federated learning algorithm can be studied in more depth. The algorithm was proposed for training neural networks trough backpropagation and SGD, followed by federated averaging for combining the local models. It would be an interesting study to compare different gradient descent optimization algorithms such as Adagrad, Adadelta, Adam, Momentum, RMSProp or use plain SGD with a simple learning rate annealing schedule [18]. Optimizers such as Adam or Adadelta provide adaptive learning rate for each of the parameters of the learning objective (in our case neural network weights matrices) and can potentially speed up the convergence of the local training on each client, and decrease costs of the federated learning such as communication and computational rounds. However, adaptive learning rate optimizers store different parameters about the computed gradients using backprop and use the information to produce the new update rule in each update step. This might conflict with the federated averaging, because the parameters will be calculated many times on each local client's training, compared to the centralized training where the parameters will be updated consequently during the training.

References

[1] "Fitbit Official Site for Activity Trackers and More." [Online]. Available: https://www.fitbit.com/se/home

[2] "How to use the Do Not Disturb while driving feature." [Online]. Available: https://support.apple.com/en-gb/HT208090

[3] S. Yao, S. Hu, Y. Zhao, A. Zhang, and T. F. Abdelzaher, "Deepsense: A unified deep learning framework for time-series mobile sensing data processing," *CoRR*, vol. abs/1611.01942, 2016. [Online]. Available: http://arxiv.org/abs/1611.01942

[4] F. J. Ordez and D. Roggen, "Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition," *Sensors*, vol. 16, no. 1, p. 115, Jan. 2016. [Online]. Available: http://www.mdpi.com/1424-8220/16/1/115

[5] N. Y. Hammerla, S. Halloran, and T. Ploetz, "Deep, convolutional, and recurrent models for human activity recognition using wearables," *CoRR*, vol. abs/1604.08880, 2016. [Online]. Available: http://arxiv.org/abs/1604.08880

[6] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," *CoRR*, vol. abs/1602.05629, 2016.

[7] A. Bulling, U. Blanke, and B. Schiele, "A tutorial on human activity recognition using body-worn inertial sensors," *ACM Comput. Surv.*, vol. 46, no. 3, pp. 33:1–33:33, Jan. 2014. [Online]. Available: http://doi.acm.org.focus.lib.kth.se/10.1145/2499621

[8] A. Mannini and A. M. Sabatini, "Machine learning methods for classifying human physical activity from on-body accelerometers," *Sensors*, vol. 10, no. 2, pp. 1154–1175, 2010. [Online]. Available: http://www.mdpi.com/1424-8220/10/2/1154

[9] Y. Chen and Y. Xue, "A Deep Learning Approach to Human Activity Recognition Based on Single Accelerometer," in *2015 IEEE International Conference on Systems, Man, and Cybernetics*, Oct. 2015, pp. 1488–1492.

[10] Y. Zhao, R. Yang, G. Chevalier, and M. Gong, "Deep residual bidir-lstm for human activity recognition using wearable sensors," *CoRR*, vol. abs/1708.08989, 2017. [Online]. Available: http://arxiv.org/abs/1708.08989

[11] J. Konecný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *CoRR*, vol. abs/1610.05492, 2016. [Online]. Available: http://arxiv.org/abs/1610.05492

[12] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Proceedings of the 24th International Conference on Neural Information Processing Systems*, ser. NIPS'11. USA: Curran Associates Inc., 2011, pp. 2546–2554. [Online]. Available: http://dl.acm.org/citation.cfm?id=2986459.2986743

[13] A. Stisen, H. Blunck, S. Bhattacharya, T. S. Prentow, M. B. Kjrgaard, A. Dey, T. Sonne, and M. M. Jensen, "Smart Devices Are Different: Assessing and MitigatingMobile Sensing Heterogeneities for Activity Recognition," in *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys '15. New York, NY, USA: ACM, 2015, pp. 127–140. [Online]. Available: http://doi.acm.org/10.1145/2809695.2809718

[14] A. K. Ramakrishnan, N. Z. Naqvi, D. Preuveneers, and Y. Berbers, "Federated Mobile Activity Recognition Using a Smart Service Adapter for Cloud Offloading," in *Human Centric Technology and Service in Smart Space*, ser. Lecture Notes in Electrical Engineering. Springer, Dordrecht, 2012, pp. 173–180, dOI: 10.1007/978-94-007-5086-9_23. [Online]. Available: https://link.springer.com/chapter/10.1007/978-94-007-5086-9_23

[15] V. Smith, C.-K. Chiang, M. Sanjabi, and A. Talwalkar, "Federated Multi-Task Learning," *arXiv:1705.10467 [cs, stat]*, May 2017, arXiv: 1705.10467. [Online]. Available: http://arxiv.org/abs/1705.10467

[16] A. Davide, G. Alessandro, O. Luca, P. Xavier, and R.-O. Jorge L, "A Public Domain Dataset for Human Activity Recognition Using Smartphones," 2013. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones

[17] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Sep. 2015, pp. 909–910.

[18] S. Ruder, "An overview of gradient descent optimization algorithms," *CoRR*, vol. abs/1609.04747, 2016. [Online]. Available: http://arxiv.org/abs/1609.04747