

Artifact: A Federated Learning Aggregation Algorithm for Pervasive Computing: Evaluation and Comparison

Sannara EK
Univ. Grenoble Alpes,
CNRS, Grenoble INP
LIG F-38000,
Grenoble, France
sannara.ek@gmail.com

François PORTET
Univ. Grenoble Alpes,
CNRS, Grenoble INP
LIG F-38000,
Grenoble, France
francois.portet@imag.fr

Philippe LALANDA
Univ. Grenoble Alpes,
CNRS, Grenoble INP
LIG F-38000,
Grenoble, France
philippe.lalanda@imag.fr

German VEGA
Univ. Grenoble Alpes,
CNRS, Grenoble INP
LIG F-38000,
Grenoble, France
german.vega@imag.fr

Index Terms—Federated Learning, algorithm, evaluation, Human Activity Recognition.

I. INTRODUCTION

This document provides the details and guidelines to run the federated learning experiments used in the study "A Federated Learning Aggregation Algorithm for Pervasive Computing: Evaluation and Comparison" [1]. The artifact is available in a public GitHub repository¹. Note that The centralized/local training approaches are not included in the repository.

Federated learning (FL) is a distributed meta-learning approach assuming one central server and N clients. Each client is represented by a local dataset. In FL, only models or details of models are communicated rather than actual data. The study aims at comparing 3 State-of-the-art federated learning aggregation algorithms and one new algorithm.

Our experiment uses two Human Activity Recognition datasets, the REALWORLD dataset, which can be obtained from the link², and the UCI dataset, which can be obtained from the link here³. Both the datasets are sampled at a 50hz frequency. The REALWORLD dataset was collected from 15 subjects from 7 different devices/body positions. On the other hand, the UCI dataset was data collected from the hip had a total of 30 subjects. We note that due to the small size of the UCI dataset, we uniformly partitioned the subject's data into 5 clients for our study. We used a window frame of 128 with a 50% overlap over 6 channels (3 for accelerometer and 3 for gyroscope). Table I depicts all instance amount per activity after processing the two datasets.

II. REQUIREMENTS & PREPARATIONS

The entire experiment was developed using Python 3.7 and was saved in an *ipynb* Jupyter Notebook format. For the

TABLE I: Activity count in the UCI & REALWORLD dataset

Activities	UCI	REALWORLD
Climbing Down	1406	32047
Climbing Up	1544	37520
Laying	1944	40843
Sitting	1777	40747
Standing	1906	40672
Walking	1722	41555
Jumping	N/A	6183
Running	N/A	45581

machine learning libraries, we used Tensorflow 2⁴ for all our implementations aside from FedMA which uses PyTorch⁵ as taken from the original authors⁶. The additional libraries and their exact version of which we have used can be found in the *requirements.txt* file. It is highly recommended to use the same libraries and framework version in the original implementation by running the following command in a virtual environment:

```
pip3 install -r requirements.txt
```

If using a GPU to run the experiments, please note that the PyTorch library must be explicitly chosen for the machine's specific CUDA version. Additionally, the library must also be specified if planned to run on the CPU. Additionally, if the user is running on a Windows machine, it is recommended to download and install visual studio to solve dependency requirements⁷.

A. Dataset preparations

Our study focuses only on the data stream of two inertial three-channel sensors: accelerometer and gyroscope. We take

¹PerCom2021-FL - <https://github.com/getalp/PerCom2021-FL>

²REALWORLD - https://sensor.informatik.uni-mannheim.de/#dataset_realworld

³UCI Dataset - <https://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones>

⁴TensorFlow - <https://www.tensorflow.org>

⁵PyTorch - <https://pytorch.org>

⁶FedMA - <https://github.com/IBM/FedMA>

⁷Visual Studio - <https://visualstudio.microsoft.com>

only the raw data of the two the dataset and apply channel-wise z-normalization.

Run the **DATA_REALWORLD_SPLITSUB.ipynb**, and **DATA_UCI.ipynb** files to download, process, and the REALWORLD and UCI dataset, respectively. The datasets post-processing result can then be found in the *datasetStandardized* folder with a sub-folder for each respective dataset.

Specifically for the REALWORLD dataset, each participant's individual dataset is separated by a number 1-15. On the other hand, the UCI dataset is uniformly partitioned into clients later on in the training implementations.

III. IMPLEMENTATION

There are three notebooks for the four FL aggregation techniques (FedAvg, FedPer, FedMA, and FedDist). In each notebook's third cell, there is an abundance of setting choices to control the training conditions directly. We only recommend changing the `os.environ["CUDA_VISIBLE_DEVICES"]` parameter to specify the GPU for training. Set this parameter to "-1" to utilize the CPU for training instead.

The FedAvg & FedPer implementations can be found in **FedAvg_FedPer.ipynb**. In this file, it is crucial to explicitly choose the algorithm that is desired to run by setting the *algorithm* parameter to "FEDAVG" or "FEDPER" as shown below with FedAvg:

```
algorithm = "FEDAVG"
```

There are two pre-configured models in this experiment. A two layered Convolutional Neural Network (196 filters of 1x16 filters in the convolutional layer followed 1024 neurons in the dense layer) and a two layered Dense Neural Network (400 Neurons in the first dense layer followed by 100 Neurons in the second layer). The neural network model in the experiments can be modified to one's needs by searching in the notebook the cell starting with the "# initializing CNN model" comment or "# initializing DNN model" to find the codes for the model configuration. Choosing which model to use can be done in the third cell by changing the variable *modelType* to either "CNN" or "DNN".

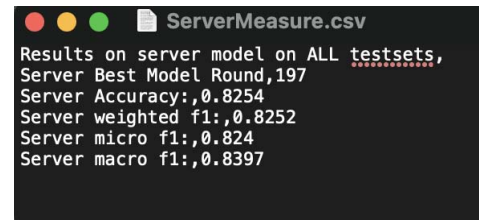
The FedMA approach is done in the **FedMA.ipynb** file. While as FedDist is found at the **FedDist.ipynb** file.

Once all configuration has been satisfactorily set, run all cells of the notebook to begin training.

The experiments were run on a Debian 4.19.132-1 version 10 having 132GB of RAM, a GeForce GTX TITAN Black 6GB GPU and an Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz for the CPU. With this configuration and the provided settings (200 Communication round and 5 Local epoch), approximately **2 days** of training time was required to finish the training of the FedAvg and FedPer implementation. FedDist and FedMA, being the longest, required **4-5 days** of training time.

IV. RESULTS

With the default environment set up, once training is completed, the best server model is saved in a *.h5* format in a



```
ServerMeasure.csv
Results on server model on ALL testsets,
Server Best Model Round,197
Server Accuracy:,0.8254
Server weighted f1:,0.8252
Server micro f1:,0.824
Server macro f1:,0.8397
```

Fig. 1: Sample of the ServerMeasurer.csv file

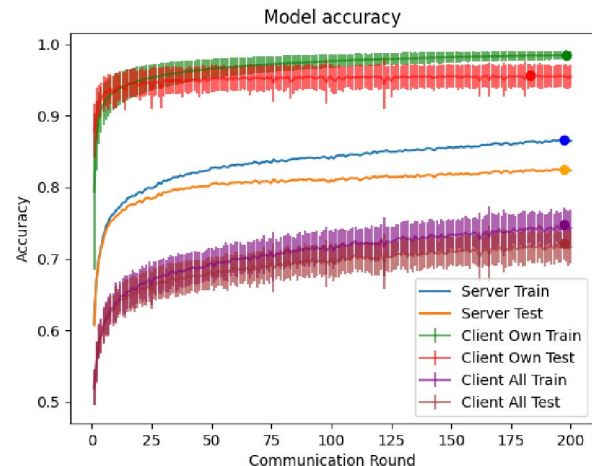


Fig. 2: Sample of the generated accuracy chart

generated *models* folder. The model shape can be visualized by opening the *.h5* file with the **netron**⁸ ML model viewer or can be loaded for further training in another instance.

Upon the finish of a training, statistical measurements and charts will be generated from within the same notebook for the specified algorithm. Figure 2 shows the charts that will be automatically generated. The recorded training statistics for each communication round, such as the accuracy and loss of the clients model and server model, are stored in the *trainingStats* folder. The training statistics are stored in a *.hkl* file format, where we have provided additional notebooks stored in the *interpret_notebook* folder to read these files. The algorithm specific notebook for the results must be moved to the local directory of the generated results folder when a training is finished to function.

The results regarding the best **Global accuracy** and the detail of the server model can be found on the generated **ServerMeasure.csv** file, as shown in figure 1. Results for the **Personalization accuracy** can be found in the *individualClientsMeasure.csv* file and finally the **Generalization accuracy** can be found in the *AllClientsMeasure.csv* file.

REFERENCES

- [1] S. Ek, F. Portet, P. Lalanda, and G. Vega, "A federated learning aggregation algorithm for pervasive computing: Evaluation and comparison," in *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2021.

⁸netron - <https://github.com/lutzroeder/netron>